An Evaluation of the Suitability of Commercially Available Sensors for Use in a Virtual Reality Prosthetic Arm Motion Tracking Device

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Electrical Engineering
University of Saskatchewan
Saskatoon, Saskatchewan, Canada

By
Jonathan Churko

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical Engineering
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

The loss of a hand or arm is a devastating life event that results in many months of healing and challenging rehabilitation. Technology has allowed the development of an electronic replacement for a lost limb but similar advancements in therapy have not occurred. The situation is made more challenging because people with amputations often do not live near specialized rehabilitation centres. As a result, delays in therapy can worsen common complications like nerve pain and joint stiffness. For children born without a limb, poor compliance with the use of their prosthesis leads to delays in therapy and may affect their development. In many parts of the world, amputation rehabilitation does not exist. Fortunately, we live in an age where advances in technology and engineering can help solve these problems. Virtual reality creates a simulated world or environment through computer animation much like what is seen in modern video games.

An experienced team of rehabilitation doctors, therapists, engineers and computer scientists are required to realize a system such as this. A person with an amputation will be taught to control objects in the virtual world by wearing a modified electronic prosthesis. Using computers, it will be possible to analyze his or her movements within the virtual world and improve the wearer's skills. The goals of this system include making the system portable and internet compatible so that people living in remote areas can also receive therapy. The novel approach of using virtual reality to rehabilitate people with upper limb amputations will help them return to normal activities by providing modern and appropriate rehabilitation, reducing medical complications, improving motivation (via gaming modules), advancing health care technology and reducing health care costs.

The use of virtual reality technology in the field of amputee rehabilitation is in its earliest stages of development world wide. A virtual environment (VE) will facilitate the early rehabilitation of a patient before they are clinically ready to be fitted with an actual prosthesis. In order to create a successful virtual reality rehabilitation system such as this, an accurate method of tracking the arm in real-time is necessary. A linear displacement sensor and a microelectromechanical system (MEMS) inertial measurement unit (IMU) were used to create a device for capturing the motion of a user's movement with the intent that the data provided by the device be used along with a VE as a virtual rehabilitation tool for new upper extremity amputation patients. This thesis focuses on the design and testing of this motion capture device in order to determine the suitability of current commercially available sensing components as used in this system. Success will be defined by the delivery of accurate position and orientation data from the device so that that data can be used in a virtual environment. Test results show that with current MEMS sensors, the error introduced by double integrating acceleration data is too significant to make an IMU an acceptable choice for position tracking. However, the device designed here has proven to be an excellent cable emulator, and would be well suited if used as an orientation tracker.

Acknowledgements

I would like to express my gratitude to my supervisors Dr. Anh van Dinh and Dr. Gary Linassi. Thank you so much for your guidance, trust and support throughout my post-graduate studies. Your kindness and understanding will never be forgotten.

I would also like to thank to Dr. Aryan Saadat Mehr for keeping me on track when I began to stray off course.

Thank you to my friends, who kept me sane and provided me with advice, laughter and friendship every time that I needed them.

Finally and most importantly, I would like to say thank you to my parents Mark and Valerie, my brother Jared and my sister Jocelyn. You believed in me even when I didn't. The positive effect that your generosity, love, and encouragement has on me cannot be overstated. I love you.

This thesis is dedicated to my parents. Without your unending love and support, this would never have been possible.

Contents

List of Tables

List of Figures

List of Abbreviations

VR - Virtual Reality
VE - Virtual Environment
SiP - Simulated Prosthesis
MEMS - Microelectromechanical System
EM - Electromagnetic
INS - Inertial Navigation System
DSP - Digital Signal Processing
EEPROM - Electronically-Erasable Programmable Read-Only Memory
PC - Personal Computer
UART - Universal Asynchronous Receiver Transmitter
SPI - Serial Peripheral Interface
ADC - Analog-to-Digital Converter
6DOF - Six-Degree-Of-Freedom
LSB - Least Significant Bit
MSB - Most Significant Bit

List of Variables

Chapter 1

Introduction

This chapter identifies the need for the improvement of the tools used by practitioners and patients during upper extremity amputation rehabilitation and explores some of the research currently being done in rehabilitation.

1.1 Research Objective

To date, no virtual training environment for the rehabilitation of new amputees has been designed. It is proposed that a virtual reality prosthetic tracking device implemented into a specialized virtual environment as suggested in this thesis would reduce the time that a new amputee must spend in the hospital since the patient can start learning to use their prosthesis earlier than if they had to wait for a real world prosthetic device. Earlier training would hasten a patient's return to society and resumption of activities of daily living. An expedited return to daily life would benefit the patient by more quickly restoring his or her self-confidence and self-worth. The cost of treating the patient would be reduced because not only would the length of the patient's stay in the hospital be minimized, but a therapist would not be required to be present for every training exercise. The virtual environment would give the patient feedback rather than relying on the doctor to watch the movements. This would allow the doctor to treat more than one patient at a time and let a patient practice the training exercises when there is no doctor present. Since a doctor could follow a patient's movement on a computer screen and give feedback without being present in the room, a system such as the one proposed would be an excellent way for rehabilitation doctors to administer treatment to patients in remote locations, effectively increasing the rehabilitation coverage of a hospital or rehabilitation therapist.

The goal of this thesis is to determine whether current commercial sensing devices are suitable for use in a motion tracking device designed specifically for tracking the position and orientation of an upper extremity amputee's arm during rehabilitation exercises.

1.2 Thesis Outline

This thesis explores whether a motion capture system implemented using inertial navigation can be used as a prosthetic tracking device in a virtual environment (VE). It focuses on a method of using three accelerometers

and three gyroscopes to track the movement of an arm so that those movements can be displayed in a VE for the purposes of rehabilitation. The first chapter discusses the need for such a device and the motivation behind the project. The second chapter provides an examination of current methods an technologies used in motion trackers. After this, the methodology and design of the device is discussed. A method of calibrating the system is provided and how the data is used to provide a measurement of the position and orientation of the device is explained. The results chapter explains the experimental setup and an analysis of the validity of these algorithms and the accuracy of the system is performed. Then, there is a discussion on significance of the results and how the affect the final product. Finally, there is a summary of the suitability of the system for the purposes of tracking an arm in a VE when used for patient rehabilitation, and a guide to the next steps which should be taken in order to continue this research.

1.3 Motivation and Design Requirements

The loss of a hand or arm is a devastating event in a person's life, after which there are many steps toward recovery. It affects every aspect of life and any means to diminish its impact and return a patient to his or her desired level of independence is worth developing. As Childress stated in 1985, "adequate replacement of the human hand and arm is one of the most difficult problems facing medical technology" [1]. Since it is not mandatory to report amputation incidence data in Canada, it is difficult to know the incidence or prevalence of upper extremity amputations in Canada. In the US, however, the Amputee Coalition of America indicates that there are approximately 1.28 million amputees with approximately 50,000 new amputations performed annually; 10% of those being upper limb amputations [2].

Modern technology has allowed the development of electronic limb replacements but similar advances in therapy has not occurred. Providing therapy for new amputees is complicated by the fact that many people do not live near the specialized rehabilitation centres currently required for proper treatment. This can cause therapy delays which can worsen common complications such as nerve pain and stiffness. The role of the health care practitioner is to use rehabilitation techniques to improve the patient's quality of life and to maximize an individual's mobility while attempting to return the person to regular day-to-day activities as soon as possible. The approach to rehabilitating a person with an upper extremity amputation has not changed much from the face-to-face therapy practises used for many years. There are nine phases of amputee rehabilitation: preoperative, amputation surgery, acute post-surgical, pre-prosthetic, prosthetic prescription and fabrication, prosthetic training, community integration, vocational rehabilitation, and follow-up [3]. The rehabilitation professional is involved in all of these phases, but a large amount of the rehabilitation time is spent in the pre-prosthetic, prosthetic prescription and fabrication, and prosthetic training phases. The pre-prosthetic phase is when the patient begins to gain limb control again after the amputation. Once the wound has sufficiently healed and the swelling has sufficiently subsided, the patient can get a prosthesis fabricated and fitted, after which they can begin training to use it.

During this training phase, the patient learns to control and use their new prosthesis, so that they can be reintegrated into the community and resume daily life activities. The patient works with a therapist in a specialized rehabilitation centre where they perform tasks and exercises designed to restore motion and teach skills required for proper prosthetic device operation. These rehabilitation centres have limited space and are often booked months in advance, resulting in an increase in total rehabilitation time during the training phase. As well as overbooked rehabilitation centres, another factor increasing training time is the limited number of rehabilitation professionals and therapy providers. Many smaller communities do not have the resources to provide specialized services and the patient ends up being required to drive long distances to receive treatment. Remote communities with limited populations or limited access and small farming communities are certainly affected by this, as are military personnel overseas. Many of the amputations today are the result of military action in war zones [4].

A virtual environment (VE) using virtual reality would allow a patient to begin learning how to use a prosthetic device before being able to wear an actual prosthesis. The rehabilitation therapist would also benefit from this virtual environment because the environment may be programmed to provide feedback on the patients movements and give simple instructions to the patient without requiring the therapist to intervene. There have been a number of virtual environments such as this that have been designed for other rehabilitation practises; many of them for stroke rehabilitation [5–7].

A VE would also facilitate the delivery of amputee rehabilitation over the internet thereby providing therapy to patients in remote locations such as northern communities or war conflict areas who would otherwise be underserved. Since the environment would most likely be used in the absence of technical assistance, the user must be able to set up the system with minimal configuration required. The system must also be easily transported and capable of being shipped long distances. For these reasons, an inertial position tracking system was chosen. A microelectromechanical system (MEMS) inertial tracking device is small enough to be comfortably worn and easily shipped to remote or dangerous locations. Current MEMS devices which measure rotation and acceleration are much more accurate and physically robust than their counterparts of even just a few years ago, making them good candidates for a device which will be shipped long distances and worn on the arm. Furthermore, calibration of the device can be done automatically with software that requires that the user only wear the device and run the program. No complicated or specialized calibration equipment or training are required for setup and use. This thesis will focus on the design and testing of this tracking device applied to a virtual reality training environment for upper extremity amputees.

Chapter 2

Methods and Technologies In Use Today

This chapter outlines some of the virtual reality devices used today by doctors in physical medicine and rehabilitation. It then goes on to discuss some of the methods used for tracking objects in the real world so that they can be used in a virtual reality system and displayed in a virtual environment. Mathematical algorithms for converting sensor data to usable forms are also examined here.

2.1 Current Rehabilitation Methods

Virtual reality (VR) is beginning to be used as a training and therapy tool for rehabilitation in many situations. There has been research applying VR to stroke rehabilitation [6, 8, 9], spinal cord injury [10], vestibular dysfunction [11], multiple sclerosis [12] and other injuries and dysfunctions.

Kuttuva and others at Rutgers University in New Jersey have developed a VR system for upper extremity rehabilitation called the Rutgers Arm [6]. This system is used to help a patient regain the use of their arm after a stroke and consists of a dual processor PC, a Polhemus Fastrak tracker, a low friction table and armrest, and an internet connection for telerehabilitation. According to Kuttuva, tests with a chronic stroke patient showed that the Rutgers Arm was successful in improving arm control and shoulder range of motion and that these gains were maintained a week later. The patient places their arm on the armrest and moves it along the low-friction table while the tracker near the wrist sends movement data to the PC. Game-type exercises realized with Java3D technology give the patient a number of exercises to do and give feedback based on what the user is doing compared to what the user should be doing. The system also includes a microphone and a speaker so that a therapist can communicate with the patient remotely. This system requires a significant amount of hardware and is limited in that the patient must be sitting with their arm on a table during the exercises. Polhemus Fastrak tracking systems use electromagnetics to track movement, which requires a special room configuration and requires that there are no sources of interference, such as ferric materials, in the vicinity.

A VR system has been developed by Kizony et al. in an attempt to help patients improve balance after a spinal cord injury [10]. Initial research done by Kizony et al. with the balance training environments found that "The positive responses to the experience as well as the expressions of interest in having additional sessions with the system suggest that this modality may increase motivation for therapy" [10]. With the

balance system, patients were given three environments in which to do the exercises given at three different difficulty levels.

Research done by Yoram Baram and Ariel Miller found that multiple sclerosis patients' walking abilities (speed and stride length) were improved by using virtual reality visual-feedback cues [12]. Baram and Miller used a head-mounted device attached to a pair of eyeglasses. The head-mounted display showed a virtual checkerboard floor which responded dynamically to the patient's movements. The patient was required to walk a track first without wearing the device and then wearing the device both with the device turned off with no visual cues and on with the device providing visual cues. After using the device, the patient was required to walk the track again without the device for the purposes of studying short term effects of the visual cues. Baram and Millar found that the average base walking speed and the average base stride length both improved and found that patients had more confidence in their movements after using the device.

There is currently no VR device which is designed for training amputees in the use of their new prosthesis, though the findings of each of the above discussed projects suggests that virtual reality may be useful in providing therapy in such a rehabilitation setting.

2.2 Tracking Methods

For a virtual prosthetic tracking device to be useful, it must meet a number of requirements. It must first be able to accurately track the arm's position in space. Secondly, it must be able to map the orientation of the arm at that position in space. These measurements must be sufficiently accurate such that any discernible movement by the user will result in that same movement shown in the virtual environment. There are a number of methods available for use when tracking an arm for display in a virtual environment, each with their own benefits and drawbacks.

2.2.1 Optical Trackers

Many motion capture (mocap) systems make use of optics to follow an object's movement. There are generally two types of optical motion capture systems: markered systems and markerless systems. Markered systems can utilize passive markers or active markers, or a combination of the two.

Passive markers are usually reflective objects worn either directly on the skin or worn on a spandex bodysuit. A light is shone at the reflectors and then a number of cameras are used to capture the reflected light. The images captured by the cameras are used to calculate the position of each marker. Due to the availability of very high resolution cameras, extremely high precision position and motion data can be captured in this manner. Retroreflective mesh suits have been used [13] with high precision cameras to provide very high precision 3D modelling and motion tracking. There is a tradeoff between precision and speed when using these types of systems since the higher the precision, the more processing involved. Passive markers are seldom used for real-time systems because computations are needed to first identify each

marker before the movement of that marker can be found and these computations can be very time intensive increasing the latency of the systems. Systems such as those devised by Yamane, Kuroda and Nakamura [14] combine high-precision camera systems with high-speed camera systems increasing the accuracy of the high-speed systems while only slightly increasing the time required for processing when compared to other passive marker systems.

Active markers generate their own light which is then captured by cameras in the same manner as the passive systems. These markers improve the passive systems because the active markers can be detected from a much further distance than the passive ones and the active markers can be turned on in a sequence, allowing the system to much more quickly identify which marker is which. The cameras in such sequential systems must capture images faster than a system which lights all markers at once, though if many colors of marker lights are used, the number of image captures necessary is decreased by a factor of the number of the number of colors used. Using many different colors of marker like this also increases the precision of the data because occluded markers can be more easily interpolated. Knowing which color of marker is missing makes it easy to identify which is the missing marker and where that marker should be [15]. Some of the pitfalls to these types of active markered systems are that the cameras and the sensors become expensive. To capture the same number of markers, the cameras must be faster in these sequential and colored systems than in their passive counterparts and in the case of the colored systems, the cameras cannot be monochromatic. The markers themselves must either be wired or individually battery powered and must have some sort of control mechanism. All of these requirements increase the complexity of the system and the expense of the components.

Markerless optical systems (sometimes also called vision-based systems) provide a solution where motion is captured using multiple cameras to video an object's movement [16]. These videos are then analyzed using any of a number of pose estimation and tracking techniques like motion vector analysis such as is used in MPEG encoding or like those outlined by Wang, Hu, and Tan [17]. The ability to track an object without requiring any markers makes applications such as motion tracking security systems which respond only to movement of one type of object possible. For example, if a human is detected, an alarm is sounded, while if a dog is detected, no alarm is raised. Motion tracking using only vision-based algorithms has a similar tradeoff to that of tracking using the other camera based algorithms, in that an increase in accuracy requires an increase in computational power. In order to accurately track an object in real-time for use in a VE, a very powerful computer is necessary. The requirement for such a powerful computer and for multiple cameras (many times more than 8) limits the portability of these types of tracking systems and limits the installations of such a system to a static location.

Any type of optical tracking system requires a significant amount of processing which limits the number of real-time applications. These solutions also generally require a significant amount of setup and calibration, usually to the point where they are installed in special studios where they can be permanently located. The high precision (and high cost) of these systems make them very well suited for applications such as cinematic

capture for use in films or video games, where post-processing can be extensively applied and the cost is not prohibitive due to the high monetary returns garnered by the studios upon the film or game's release.

2.2.2 Electromechanical Trackers

If a limited range of motion is acceptable, then electromechanical motion tracking systems are available. These systems use actuators and physical position sensors such as potentiometers or other variable resistance devices to track movement. Since the object being tracked is connected to a physical machine, motion is limited to the range of the machine. These types of trackers are useful for orientation sensing, where they can be applied to things such as joints. For position sensing, however, they are less useful because they require a point of reference to always be attached. For example, it would be easy to use an electromechanical sensor to track the angle at which an elbow is bent, but it would be less practical to track a person's location in a room with one of these sensor setups. Electromechanical sensors are often used in robotics because they allow high-precision position or angle sensing and correction when used on parts which have a limited range of movement. They are used less often in human tracking scenarios due to the number of complex movements which the human body can perform, making the number of sensors required cumbersome and prohibitive. Another downfall to electromechanical sensors is that the mechanical devices required to be worn are often uncomfortable, heavy, unsightly, awkward, and difficult to wear. Since electromechanical sensors are quite accurate, they are still used in applications where comfort is not an issue but accuracy is, such as skeletal modelling for use in virtual computer applications. An example of a commercial electromechanical system is the Gypsy 6$^{\text{TM}}$Motion Capture System by MetaMotion [18].

2.2.3 Magnetic Trackers

Electromagnetic (EM) motion capture systems use a magnetic field generator and magnetic field sensors to accurately track position and orientation of the sensors. Some of these systems use pulsed DC magnetic fields but many EM trackers use AC fields because the DC fields are affected by the earth's magnetic field. In these systems, the sensors usually consist of three orthogonal coils. When the coils are located in the point-source generated magnetic field, a current is induced in the coils. Based on the amount of current induced in each of the coils, the position and orientation of the sensor can be found. If there are ferrous materials inside the generated magnetic field, a current will be induced in the non-sensor material, creating another magnetic field [19]. This extra generated field will cause erroneous measurements by the sensors and any other external magnetic field will interfere with the sensors' measurements in the same manner. While some of these interferences can be fixed algorithmically, the fix will useless if the interfering materials or magnetic fields change.

Polhemus [20] is one of the leading companies in motion capture technology. They specialize in AC electromagnetic tracking systems. These systems provide very low latency and high accuracy when used in controlled stable environments where there are no magnetic interferences or disturbances. While these

systems are perfect for a permanent installation where the surrounding environment can be controlled, they are still highly susceptible to interference from external magnetic fields and ferrous materials. This makes an EM tracking system unsuitable for mobile use where the user may be unaware of any interfering objects.

2.2.4 Inertial Trackers

Inertial motion capture systems use accelerometers and gyroscopes to track the movement of an object. A 6-degree-of-freedom (6DOF) system requires three orthogonal gyroscopes and three orthogonal accelerometers. A gyroscope measures the rotational velocity about an axis and an accelerometer measures the linear acceleration along its sensitive axis. Based on an initial position, the position of the object in space can be found by double integrating the accelerations that the object has undergone. The gyroscopes give rotational velocities in three dimensions and can be single integrated to determine the orientation of the object.

Since the position and the orientation of the object in an inertial navigation system (INS) are found by double integrating acceleration and single integrating angular velocity data, these systems are very susceptible to drift due to sensor errors. Because the errors are summed, the longer the system runs, the more inaccurate the calculated data will be. Many but not all of these errors can be modelled and accurately removed. The more errors are accumulated, the faster the system will lose accuracy. The increasing error in INS's requires that these systems are recalibrated on a regular basis. Most INS implementations are supplemented by an absolute tracking device such as GPS. The INS is used for constant position calculation, while the GPS is used to periodically correct the INS. The reasons that GPS is not used exclusively are that GPS position updates are slow, the resolution of GPS navigation is too low to be used for small-range position tracking and GPS cannot be used without line-of-sight to the GPS satellite. GPS cannot be used indoors or in tunnels, etc.

Unlike most of the other motion capture methods, there is no need for a transmitter and a sensor to be set up. All of the motion data can be found using sensors alone. Advances in microelectromechanical systems (MEMS) have provided the opportunity for the creation of miniature accelerometers and gyroscopes which are small enough to fit on a printed circuit board (PCB), making wearable wireless inertial sensors possible. Because there is no requirement for a transmitter in inertial systems and due to the recent miniaturization of the sensors involved, inertial motion capture systems lend themselves well to portability. It is for this reason that INS systems have been used for terrestrial navigation in aeroplanes, boats and land vehicles since the 1950's.

2.3 Inertial Tracking

Of all of the tracking systems discussed in section 2.2, the most suitable for a virtual reality-based prosthetic arm training device is an inertial tracker. Inertial tracking can be accomplished using very lightweight MEMS accelerometers and gyroscopes, allowing the device to be worn comfortably on the arm. Since inertial trackers

can be very small and do not require any receivers or specialized setup to be performed by the user like optical and magnetic trackers do, they lend themselves well to portability. They can easily be set up in any location by the user with minimal instruction.

2.3.1 Inertial Measurement Units

Inertial tracking and navigation has been used in many applications such as those involving aircraft, submarines, missiles and spacecraft beginning in the 1940's. Since they were first developed, inertial measurements units (IMUs) have become smaller and more accurate than the original designs. For example, the IMUs used in the original Snark missile weighed a few hundred pounds, whereas an IMU today with the same accuracy weighs less than a few pounds [21]. In recent years, solid-state IMUs have been developed using MEMS technology allowing sizes of less than a cubic inch and weighing only ounces, albeit with slightly lower accuracy.

A basic IMU consists of a number of accelerometers and gyroscopes. Other aiding sensors such as magnetometers or GPS recovers may be included but an IMU will almost always include accelerometers and gyroscopes. For a six degree of freedom (6DOF) sensor, three accelerometers and three gyroscopes are needed. The three accelerometers are arranged in an orthogonal fashion and the three gyroscopes are arranged coaxially with the three accelerometers. Any type of accelerometer or gyroscope can be used, depending on the size, cost or accuracy requirements.

For applications involving navigation of aircraft, spacecraft or land vehicles, the accuracy of the sensor is more important than the weight of the sensor. In these applications, the sensors must be operated for long periods of time (on the order of hours) without any recalibration. To get the type of accuracy required for navigation, the sensors must be very accurately designed with extremely tight tolerances. The only sensors available with this type of accuracy today weigh a few pounds or more. This is generally not a problem for vehicular navigation, because adding a couple of extra pounds to the vehicle doesn't change the operation.

When the weight of the IMU is an issue, such as in remote-controlled airplanes or helicopters, MEMS sensors are usually used. These sensors are very light (usually on the order of milligrams), but have lower accuracy than their heavier aircraft or military grade counterparts. Because of the lower accuracy of these sensors, they are not suitable for motion tracking for long periods of time, but they are very useful for short-term navigation.

A virtual reality-based prosthetic arm training device does not need to be operated for long periods of time. The exercises and motions which a patient would be required to perform generally take less than a minute each and the device can be recalibrated after each movement. Even though sensors with the greatest accuracy would be preferred, the accuracy of the sensor is not as crucial for these short durations as it would be if the device was required to operate for long periods of time. Since the device must be worn by a patient with a fresh wound, it must be small in size and lightweight. A MEMS-based IMU is perfectly suited for such an application because it is very lightweight and still reasonably accurate.

MEMS Accelerometers A MEMS accelerometer measures acceleration along its sensitive axis with respect to free-fall. This is accomplished using a capacitive sensor that includes a mass which is suspended by a spring mechanism. This mass is constrained to movement along a single axis. When the mass moves, the spring restricts the mass' movement. The distance that the mass moves is proportional to the acceleration which the device experiences along the sensitive axis. This distance is measured by a capacitive sensor which generates a change in electrical signal based on the distance the mass has moved.

An image of a typical MEMS sensor is given in Figure 2.1. The acceleration measured by one of these capacitive MEMS accelerometers is relative to free-fall. This means that the accelerometer measures gravity as well as any other external forces acting upon the sensor. The measurement given by the sensor is the sum of the gravitational force acting on the sensor, other external forces acting on the sensor, and some noise due to the sensor's inaccuracy and physical characteristics. Equation 2.1 shows the measurement, a_m, of a single axis accelerometer where a_e is the acceleration due to external forces, g is the acceleration due to the force of gravity, and u is the noise in the measurement.

$$a_m = a_e + g + u. \tag{2.1}$$

The noise in the measurement can be approximately determined using statistical models and the physical characteristics of the sensor — these characteristics can be obtained by physically testing the sensor or from the sensor's documentation. In some applications the measurement of gravity is desirable, but for inertial tracking, the gravity component in the sensed signal must be removed so that only the forces involved in the movement of the sensor are present. Unless the orientation of the accelerometer with respect to the direction of gravity is known, it is impossible to know what part of the measurement is due to gravity and how much is due to the external force (unless of course the external force is already known). Gyroscopes are used to determine which part of the acceleration is gravity and which part is an externally applied force.

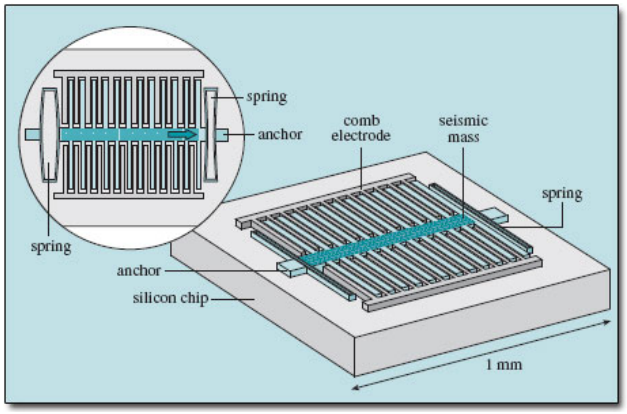

Figure 2.1: Typical MEMS Accelerometer

Source: http://openlearn.open.ac.uk/mod/resource/view.php?id=216212 Accessed Jan. 2009

MEMS Gyroscopes MEMS gyroscopes measure rotational velocity around a sensitive axis. Instead of using the spinning disk like most physical gyroscopes, the MEMS gyroscope (sometimes called a Vibratory Structure Gyroscope [22]) is created using a principle similar to the Foucault Pendulum [23]. A simple representation of the construction of a MEMS gyroscope is given in Figure 2.2. The MEMS gyroscope has a small proof mass which is attached to an inner frame by springs. The inner frame is then connected with springs to the outer frame. The mass springs are perpendicular to the frame springs. The proof mass has two perpendicular degrees of freedom. It is driven to vibrate at high frequencies in the mass drive direction (up and down in the illustration). As the gyroscope body rotates, inertia causes the mass to continue vibrating along the same path, even though the frames of the gyroscope have rotated around the mass. Before the rotation, all of the vibration is in the mass drive direction. During the rotation, the orientation of the frames have changed, but the actual direction of the vibration hasn't. The proof mass is now vibrating along an axis which is partially in the sense direction and partially in the mass drive direction with respect to the frames. By using the rotational velocity of the sensor to track orientation, the externally applied acceleration can be extracted from the accelerometer and the position of the sensor in space can be found.

2.3.2 Coordinate Systems

In order to track an object in space, a point of reference and a coordinate system at that point must be defined. The position and orientation of that object can be described based on this point. This is the basis of any computerized tracking system.

Motion tracking systems meant for tracking airplanes or land vehicles generally use a coordinate system which is defined with respect to the center of the earth. An earth-centred, earth-fixed coordinate (ECEF) system is one such coordinate system. In an ECEF coordinate system, the origin of the system is at the earth's center of mass. The z-axis is the axis of rotation of the earth with north positive, the x-axis points from the origin through the intersection of the equator and the prime meridian and the y-axis completes the right-handed coordinate system. In a coordinate system such as this, the gravity vector always points toward

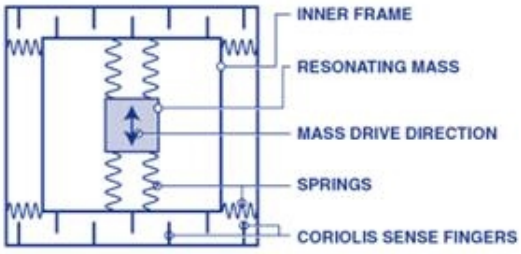

Figure 2.2: Simplified MEMS Gyroscope

Source: http://knol.google.com/k/hamid-vajihollahi/mems/2g44ddbae4k1v/2 Accessed Jan. 2009

the origin, but the relative orientation of the gravity vector differs depending on the current location. This coordinate system is useful when tracking an object's location on earth, but has little use in an application where the object doesn't move significantly with respect to an object with a size such as that of the earth.

Another coordinate system most often used in terrestrial navigation is the north, east, down (NED) coordinate system. In an NED coordinate system, the z-axis points in the direction of gravity (down), the x-axis points along the line which is tangent to the earth and pointing in the direction of the north pole, and the y-axis points along the line which is tangent to the earth and pointing towards the east. The gravity vector in this system always points in the down direction and thus, always in the z-direction. This type of coordinate system is useful when it is required that an object's absolute position on earth be known.

A simpler local fixed coordinate system which is similar to the north, east, down system can be defined relative to the immediate surroundings based on an initial location. In this system, the z-axis points in the direction opposite that of gravity (up), the y-axis points in the forward direction from the starting point, and the x-axis points to the right of the starting location. This is similar to the north, east, down coordinate system except that it is unnecessary to define the starting x- and y- axes in any specific direction other than being normal to the gravity vector and each other. This type of coordinate system is useful when it is only necessary to track the object with respect to its immediate surroundings and not necessarily with respect to the earth.

A virtual environment will have its own coordinate system and all objects in the environment will be defined with respect to that reference. The purpose of a virtual environment is to simulate the real world in a computer. For this reason, it seems reasonable that a coordinate system in the virtual world would be chosen to coincide with a real world coordinate system. This coordinate system should be fixed and referenced to an initial position and orientation. The most intuitive coordinate system for a virtual training environment is where the z-axis points in the direction of gravity (up positive), the y-axis points in the forward-backward direction (forward positive), and the x-axis points in the left-right direction (right positive) from the starting point in the same manner as described above. This starting point can be defined by a position on a table, the middle of a room or any other arbitrary point in nearby space.

Since neither the room that the user is standing in, nor the room displayed in the virtual environment moves with respect to the initial location, it is easy to display real world movements in the virtual environment. Those movements, however, must be defined with respect to that initial location in the local fixed reference system. An inertial measurement unit measures movement in its own frame of reference and as it rotates, so does that frame of reference. Since the frame of reference for the IMU's measurements moves, another coordinate system (the sensor coordinate system) must be defined to coincide with the IMU's sensitive axes. This means that the measurements taken by the IMU are valid in the sensor coordinate system, but not in the local fixed coordinate system. In order for the measurements to be useful in the local fixed coordinate system, the measurements must be transformed from the sensor coordinate system to the local fixed coordinate system.

2.3.3 Rotation Representation

Once the fixed and the sensor coordinate systems are chosen, there must be a way to relate one coordinate system's orientation to the other's. There are many ways of describing one coordinate system's orientation with respect to another and each have their own advantages and disadvantages. A few of the methods used for rotating a rigid body's orientation from one coordinate system to another are the Direction Cosine Matrix (DCM), axis and angle rotation, Euler angles and Quaternions.

Direction Cosine Matrix (DCM)

The DCM, sometimes known as the rotation matrix, is often used in inertial navigation systems for representing the orientation of one coordinate system with respect to another. Figure 2.3 shows two coordinate systems, xy and $x'y'$, and vector $\vec{r}$. In this figure, the angle between the x axes of the two coordinate systems is represented by ψ. In [24], Rogers shows that the relationship between the components of $\vec{r}$ in the two coordinate systems is given by

$$a_2 = \cos(\psi)a_1 + \sin(\psi)b_1$$
$$b_2 = -\sin(\psi)a_1 + \cos(\psi)b_1$$

which can be written in a matrix form as $\vec{y} = \mathbf{C}(\psi)\vec{x}$ where $\vec{y} = [\, a_2 \; b_2 \,]^T$, $\vec{x} = [\, a_1 \; b_2 \,]^T$ and the rotation matrix $\mathbf{C}(\psi)$ is

$$\mathbf{C}(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix}.$$

This rotation in two dimensions can be extended to a three-dimensional coordinate system. A three-dimensional coordinate system such as the one shown in Figure 2.4 can be rotated once around each axis in succession to perform any three dimensional rotation This is equivalent to three two-dimensional rotations.

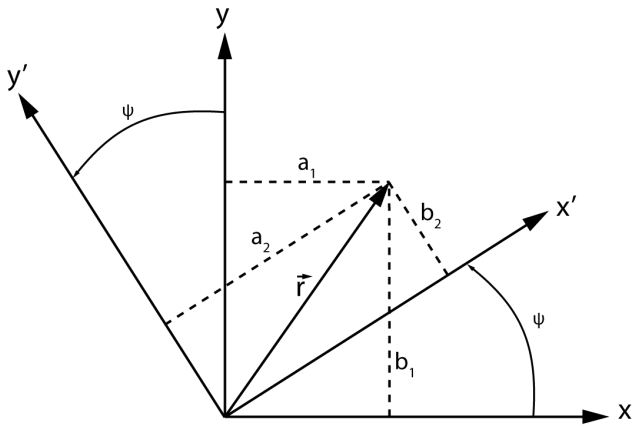

Figure 2.3: Two-dimensional coordinate frames

The two-dimensional rotation matrix shown above can be extended to the three-dimensional case as:

$$\mathbf{C}(\psi) = \begin{bmatrix} \cos(\psi) & sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

This is equivalent to a rotation around the z axis by an angle ψ with the x axis rotated toward the y axis. Similarly, rotations around the y and x axes can be accomplished by $\mathbf{C}(\theta)$ and $\mathbf{C}(\phi)$, respectively.

$$\mathbf{C}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{C}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

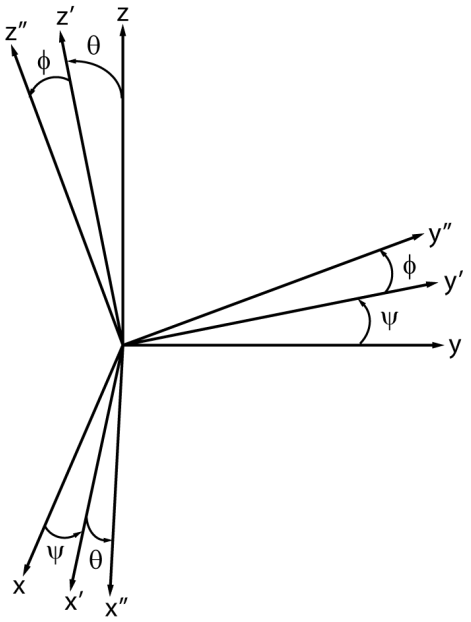

Figure 2.4: Three-dimensional coordinate frames

The multiplication of 3×3 matrices is not commutative so the order in which these rotations are applied is important. The rotations in Figure 2.4 were applied around the z axis, then the y, then the x. The order of multiplication follows the order of rotation, with the first rotation being left-multiplied and the last

rotation being right-multiplied. The resulting DCM is

$$\mathbf{C}(\phi,\theta\psi) = \mathbf{C}(\phi)\mathbf{C}(\theta)\mathbf{C}(\psi) \tag{2.2}$$

$$= \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi+\sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi+\cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi+\sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix}. \tag{2.3}$$

If a vector is represented in a sensor coordinate system and the sensor coordinate system must go through rotations ψ, θ, ϕ to be aligned with the local coordinate system, then the vector components represented in the local coordinate system can be given by

$$\vec{r}_l = \mathbf{C}_s^l(\phi,\theta,\psi)\vec{r}_s$$

where $\vec{r}_s$ is the vector represented in the sensor coordinate system, $\vec{r}_l$ is the vector represented in the local coordinate system and $\mathbf{C}_s^l(\phi,\theta,\psi)$ is the DCM which transforms the components of $\vec{r}_s$ to $\vec{r}_l$.

While many computer graphics applications use DCMs to rotate vector components from one reference frame to another, the requirement that the rotations be performed in a specific order is not optimal for a solution where the rotations are extracted from gyroscope output. Gyroscopes provide rotational velocity values of each axis independently of another axis, so the rotations are simultaneously applied, rather than sequentially. Converting the gyroscope data to Euler angles from rotational velocity is an extra step which is not necessary to be able to map the rotation of a sensor coordinate frame and as such makes the DCM a non-optimal solution for inertial tracking using gyroscopes and accelerometers.

Axis and Angle or Rotation Vector

Rather than rotating the sensor coordinate frame to the local frame by rotating about each axis sequentially, a rotation about a vector can be used to rotate the coordinate system with only one rotation necessary, rather than three. This vector is a unit vector, $\vec{v} = a\mathbf{i}+b\mathbf{j}+c\mathbf{k}$, located at the origin of the sensor coordinate frame. The angle through which the sensor coordinate frame must rotate about $\vec{v}$ to be coincident with the local coordinate frame is represented by α. John Vince shows in [25] that the rotation of a point (x, y, z) about the axis $\vec{v}$ by the angle α is

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} a^2K+\cos\alpha & abK-c\sin\alpha & acK+b\sin\alpha \\ abK+c\sin\alpha & b^2K+\cos\alpha & bcK-a\sin\alpha \\ acK-b\sin\alpha & bcK+a\sin\alpha & c_2K+\cos\alpha \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix} \tag{2.4}$$

where $K = (1-\cos\alpha)$, s denotes the point represented in the sensor coordinate frame, and l denotes the point represented in the local coordinate frame.

The axis-angle method of rotations is useful when the axis of rotation is already known, but in many cases this axis is not known and must be calculated. While successive rotations in this method are possible, they are non-intuitive and combining rotations is not directly possible with an axis-angle representation [26];

these are properties which are undesirable in inertial tracking, where rotations are applied sequentially at timesteps. Rotations with this method are not actually vectors, so vector addition of rotations is not possible. The order of rotation is important. This method requires a number of trigonometric calculations, and as such is also processing intensive. It is desired that the number of calculations be minimized, as an inertial tracking system must be able to run in real time, so a solution which does not involve trigonometric functions is required.

Quaternions

Quaternions were discovered in 1843 by William Rowan Hamilton when he was looking for a way to represent complex numbers in higher dimensions [27]. Instead of a complex number being a point on a plane, he showed that a complex number can be a point in a three-dimensional space. The quaternion is a quadruple of real numbers which takes the form

$$\mathbf{q} = [r + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}] \tag{2.5}$$

$$= [r, \vec{p}] \tag{2.6}$$

and follows the rules $i^2 = j^2 = k^2 = ijk = -1$ and

$$\begin{aligned} ij &= k & ji &= -k \\ jk &= i & kj &= -i \\ ki &= j & ik &= -j. \end{aligned}$$

As is obvious from these rules, quaternion multiplication is not commutative, $\mathbf{q}_1\mathbf{q}_2 \neq \mathbf{q}_2\mathbf{q}_1$. Addition and subtraction of quaternions, however, is commutative.

Where quaternions are really useful in designing an inertial tracker is that they make spatial rotation of a vector extremely easy. This allows rotation from the sensor frame to the local frame to be accomplished much more easily than with the other rotation representations. Vince explains in [25] that a vector rotation is accomplished with quaternions by using the equation:

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}. \tag{2.7}$$

In this equation, $\mathbf{p}$ is the quaternion to be rotated and $\mathbf{q}$ is the transforming quaternion. The quaternion $\mathbf{p}$ is converted from a position vector to a quaternion by placing the components of the position vector into the vector part of the quaternion and setting the scalar part of the quaternion to zero. If the point that the vector points to is $P(x, y, z)$ then the quaternion is $\mathbf{p} = [0 + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}]$. The quaternion $\mathbf{q}$ is created from the unit vector, $\vec{u}$, around which the rotation will take place and the angle, θ, through which the rotation will traverse.

$$\vec{u} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \tag{2.8}$$

$$\mathbf{q} = [\cos(\theta/2), \sin(\theta/2)\vec{u}] \tag{2.9}$$

The inverse of the transforming quaternion, q^{-1}, is

$$\mathbf{q}^{-1} = \frac{[\cos(\theta/2), -\sin(\theta/2)\vec{u}]}{|\mathbf{q}|^2}. \tag{2.10}$$

Where $|\mathbf{q}|$ is the magnitude of the quaternion (in this case $|\mathbf{q}| = 1$).

Multiplication of quaternions is defined as

$$\begin{aligned}\mathbf{q}_1\mathbf{q}_2 = [&(r_1r_2 - x_1x_2 - y_1y_2 - z_1z_2) + (r_1x_2 + x_1r_2 + y_1z_2 - z_1y_2)\mathbf{i} \\ &+(r_1y_2 - x_1z_2 + y_1r_2 + z_1x_2)\mathbf{j} + (r_1z_2 + x_1y_2 - y_1x_2 + z_1r_2)\mathbf{k}.\end{aligned} \tag{2.11}$$

The multiplication of quaternions doesn't require any trigonometric calculation and this, along with the fact that quaternion rotation doesn't suffer from the same gimbal lock problems that the rotation vector and DCM methods do, makes using quaternions for coordinate system rotations the most suitable of the methods described here for the types of spatial rotations required by a real time inertial tracking system.

Chapter 3

Requirement Analysis and Device Design

The design of a wearable motion tracking sensor requires both hardware and software components to be successful. This chapter will outline how the device needs to perform in order to be useful for this puropose. It will then describe the hardware which was designed for the wearable sensor, the firmware which allows the hardware to interface with a PC and the software which runs on the PC to analyse the data and convert it to usable information.

To create a virtual reality-based prosthetic arm training device, a microprocessor, a communications device and a number of sensors are used. The data from the sensors must be sent to the PC by way of the communications device where they are used to display the arm in the virtual environment. The wearable device must be small and light enough so as to not cause discomfort when worn by a person who has had a recent upper extremity amputation. It also must be placed and worn on the newly amputated limb without harming the healing incision. Since the purpose of the device is to teach a new amputee how to use a prosthesis, or to provide rehabilitation training to an amputee in remote locations, the device must resemble a real prosthetic device. To meet all of these requirements, the device must be worn just below the elbow, strapped around the forearm with the sensors mounted near the outside of the elbow.

3.1 Analysis of Required Measurements

3.1.1 Range of Motion

Since the device will be used by a new amputee, it must be lightweight and able to be worn somewhere above the amputation site. All of the movements which are required to operate a prosthetic arm must be tracked by the device. An arm can be moved anywhere in three dimensional space, with the limitation that the arm is attached to the person and therefore can only move a limited distance from the shoulder. As well as linearly moving in any direction, the arm can also be rotated about any arbitrary axis. Due to the nature of these motions, it is decided that detection of movement with six degrees of freedom (6DOF) is required. This means that linear motion must be detected along three orthogonal axes and rotational movement must be detected about these axes.

It is not sufficient just to know the types of movements which are needed to model arm movement. The maximum magnitudes and speeds of these movements must also be known. In an inertial tracker, linear

movement is found by measuring acceleration and rotational movement is found by measuring rotational velocity. In order to measure acceleration and rotational velocity, an IMU must be used. Since this thesis included building an IMU-based motion tracker, the IMU specifications needed to be chosen based on assumed maximum linear acceleration and rotational velocities and then tested after the hardware was built. Acceleration is measured in $\mathrm{m/s^2}$, or in many cases with respect to the acceleration of an object in free fall due to gravity ($9.81\,\mathrm{m/s^2}$). When acceleration is given with respect to gravity, it is given with unit g where $1g = 9.81\,\mathrm{m/s^2}$

An estimate of the average acceleration of a human arm during normal movement was made based on the idea that human movement has low acceleration when compared to a car accident. If a vehicle is travelling at 100 km/h (27.78 m/s) and gets in an accident where it takes 0.3 s for the vehicle to come to a stop, this would correspond to an acceleration of $92.6\,\mathrm{m/s^2}$, or $9.44g$. Most humans will not move with an acceleration such as that experienced in a car crash, so it is reasonable that the accelerometers will not need to be able to measure more than $10g$. Also, an average human is likely able to accelerate faster than the acceleration due to gravity in free fall, since it is possible to catch an object dropped from the air before it hits the ground. This implies that a person can accelerate faster than $1g$ ($9.81\,\mathrm{m/s^2}$). Using this information, it was assumed that a reasonable maximum linear acceleration value for normal human arm movement was $5g$ ($29.43\,\mathrm{m/s^2}$).

Rotational velocity is measured in degrees per second (°/s). In order to estimate the maximum rotational velocity of normal human movement, it was counted how many times in one second a hand could be turned from palm facing down to palm facing up and back to palm facing down. This gives an approximation of how many times a hand can be rotated through 360 degrees in one second. We experimentally found that a hand can be rotated through 360 degrees as many as ten times per second, implying rotational velocity of 3600 °/s. This was done by holding the device and rotating it as fast as possible back and forth, then analyzing the sensor data to find out how many times it had rotated in one second. The highest rotational velocity which can be measured by these sensors is ±300 °/s. Testing will show whether or not this sensor is sufficiently accurate to capture the majority of arm movements.

3.1.2 Prosthetic Arm-Specific Movements

A prosthetic arm (shown in Figure 3.1) has a terminal device on the end; usually a hook. The terminal device is operated with the shoulder opposite that of the prosthetic arm. A shoulder strap is attached to a cable which runs the length of the arm. The cable is attached to the terminal device. Since the most common terminal device is a hook, that will be the one which is used to train the new amputee. When the shoulders are flexed, the shoulder strap pulls on the cable which in turn opens up the hook. When the shoulders are relaxed, the cable puts less tension on the hook and the elastics on the hook pull it closed. Since the cable runs along a linear path through a sheath, the distance that the hook has opened can be directly measured by how far the cable has been pulled through the sheath. Measuring the linear distance that the cable has moved will give the distance that the hook has opened. A prosthetic arm's cable was measured and the full

linear range of motion of the cable is approximately four inches from hook fully closed to hook fully opened.

If 6DOF movement and the distance that the hook is opened are measured, it will be possible to model and display the prosthetic arm in a virtual environment.

3.2 Component Selection and Design

3.2.1 First Prototype

Two prototype sensors and an off-the-shelf device were used in this thesis. The first prototype was designed around a Microchip dsPIC30F6014 DSP microcontroller, the Analog Devices' iSensor ADIS16003 and ADIS16250 sensors, and a CS109AV linear displacement sensor made by MTS Sensors ®. The ADIS16003 is a dual-axis accelerometer which measures accelerations in the range of $\pm 1.7g$. The ADIS16250 is a gyroscope which measures rotations about a single axis in one of three selectable ranges: $\pm 320\,^{\circ}/s$, $\pm 160\,^{\circ}/s$ and $\pm 80\,^{\circ}/s$. Both of these sensors provide digital outputs using the serial peripheral interface (SPI) communication protocol. The CS109AV is an analog output linear displacement sensor which measures linear movement with a stroke length of 109.3mm. It is a magnetostrictive sensor which uses a magnet wrapped around a stationary waveguide element. As the magnet moves along the waveguide element, the analog output of the sensor changes proportionally with the distance that the magnet moved. The output of the sensor is ratiometric with respect to the supplied reference voltages; in this case 0 V and 5 V. The voltage output by the sensor was applied to one of the analog-to-digital converter pins (AN0) of the microcontroller, where it was converted into a digital value to be transmitted to the PC over the UART serial port.

This first prototype was intended to be a proof-of-concept device which would allow us to decide if motion could be tracked using low cost sensors without having to pay for the more expensive IMU sensors offered by Analog Devices and other manufacturers. The board had two copper layers with dimensions of 152 mm long by 83 mm wide and 38 mm tall. The size of this board was too large to wear on an arm, but was small enough to permit rough tracking movements. Figure 3.2 shows a photo of the sensor circuit board. The schematic for this device can be found in Appendix A and the PCB layout is given in Appendix B. Figure 3.3 shows a photo of the linear displacement sensor which was used to measure the distance the hook had opened.

Communication with a PC in the first prototype was accomplished using a serial cable attached between the PC and the prototype device. This meant that the device needed to be in close proximity to the PC which was receiving the data and the number and scale of the movements were limited. This was acceptable on the first device because it was only meant to be a proof-of-concept and not meant to be a final tracking device.

Once the device was built, it was found that the accelerometer and the gyroscopes would transmit data from the sensor device to the PC and that the movements were captured by the device. This design of this device however, limited it to tracking rotations about one axis and accelerations along two axes, which did

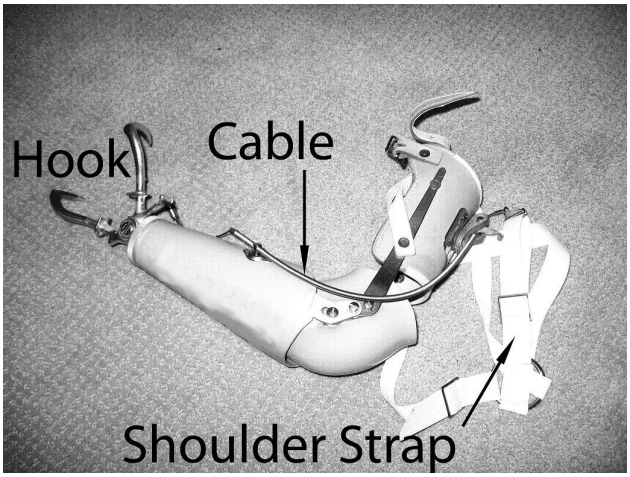

Figure 3.1: Typical Cable-Operated-Below-Elbow Prosthetic Device

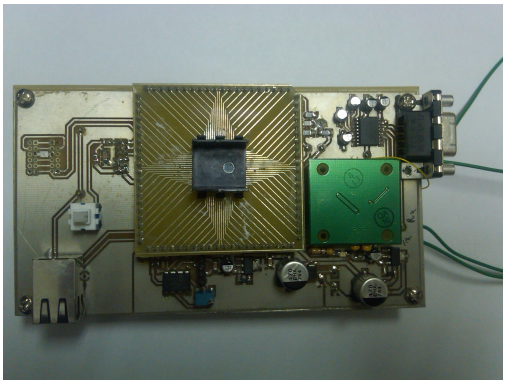

Figure 3.2: First device prototype

Figure 3.3: Linear Displacement Sensor

not permit tracking motion in three dimensions. Since the first prototype successfully captured movements, it was decided that it was possible to track movement in three dimensions using these low-cost devices.

Linear Displacement Sensor Implementation

This prototype included the linear displacement sensor which was meant for tracking the opening and closing of the prosthetic arm's terminal device (in this case a hook). The concept of using a linear displacement sensor to track the opening of the hook was based on the mechanics of how a hook opens and closes. With a typical prosthetic arm hook, there are two pincers which make up the hook. One of the pincers is stationary and held rigidly to the prosthetic. The other pincer is attached to the prosthesis by a hinge that allows the pincers to spread apart or close together. The pincers are normally held closed by tension created by elastic bands; the number of elastic bands applied to the prosthesis determines the strength of the grip created. In order to open the pincers, the wearer of the prosthesis pulls on a cable which is attached to the hinged pincer. This cable is routed up the arm and across the back through a sheathing, where it is attached to a shoulder strap. The shoulder strap is worn on the shoulder opposite to the prosthetic device. Protracting the shoulder blades causes the cable to shorten, thereby pulling on the hinged pincer. As the user pulls the cable, the pincer rotates on the hinge and the hinge opens a distance proportional to the distance that the cable has been pulled or shortened.

Since the cable moves linearly and is held in place by a sheath around the cable, it was reasonable to calculate the distance that the hook was open based on the distance the cable had been displaced. It was for this reason that a linear displacement sensor was chosen for measuring the hook opening. The Temposonics ® CS109AV by MTS Sensors ® has a stroke length (the useful span that the sensor can measure) specified as 4.311 inches (109.3mm). The cable on the test prosthetic device moves approximately four inches from fully closed to fully open. This made the CS109AV the perfect choice for this application.

In order to convert the analog output of the Temposonics ® sensor to a useful value in the microcontroller, the integrated 12-bit analog to digital converter (ADC) was used. Since the dsPIC30F6014A has an ADC which tolerates 0 V to 5 V reference voltages and the output of the Temposonics ® sensor has a range of 0 V to 5 V, only a buffer amplifier was needed to connect the sensor to the microcontroller. An Analog Devices MCP601 operational amplifier was used for this purpose. To ensure the cleanest and most stable signal possible, a separate ground plane was used for the analog section of this prototype. The exact circuit used for the analog portion of the prototype can be found in Appendix A.

An ADC requires three pins to convert an analog representation to a digital representation. Two pins are used for the reference voltages, in this case 0 V to 5 V. The third pin is used to receive the signal for conversion. The digital value given to that applied signal is ratiometric between the two reference voltages. In other words, if the reference voltages are 0 V and 5 V, when 5 V is applied to the conversion pin, the maximum digital value is given to the signal. Since the dsPIC30F6014A has a 12-bit ADC, the maximum value is $2^{12} = 4095$. If 0 V is applied, then a value of 0 will be given to the signal. Any voltage between

these values will be given a value of $d = (\frac{V_a}{5V})4095$ where V_a is the voltage applied to the conversion pin and d is the digital value given.

Once the analog signal is converted into a digital representation in the circuit, it is necessary to find a scale to convert the raw digital signal to a distance measured in the real world. This scale was determined using two methods. The full useful measurement range of the linear displacement sensor was found first. The first way the scale was determined was by placing the sensor at both ends of the full measurement range and noting the digital values produced by the sensor at these positions. The scale was then found using Equation (3.1), where x is the distance along the linear displacement sensor and d is the digital value given by the ADC. The resulting scale is given by distance/LSB.

$$\frac{x_{end} - x_{begin}}{d_{end} - d_{begin}} = scale \tag{3.1}$$

The scale found in Equation (3.1) calculates the distance the sensor's magnet has moved along the sensor's waveguide element. When the sensor is placed on the prosthetic arm device, it measures the distance the cable has moved, not the distance the hook has opened. In order to calculate the distance the hook has opened, a scale must be found between the distance the hook has opened and that which the cable has moved. Equation (3.2) is used to calculate the distance the hook has opened per distance that the cable has moved, where x is the position of the cable along the arm and h is the amount that the hook has been opened. The result, $scale_h$ is given as a proportion of distance/distance. The distance the hook opens per digital bit provided by the ADC is found by multiplying $scale * scale_h$. This gives a scale that can be directly used to calculate the distance the hook has been opened based on the output of the linear displacement sensor. The scale result found is discussed in Chapter 4.

$$\frac{h_{open}}{x_{end} - x_{begin}} = scale_h \tag{3.2}$$

Due to a successful and complete result with the linear displacement sensor in the first prototype, it was decided that it was unnecessary to implement it in the second prototype. By omitting the sensor from the second prototype, the time and cost of development for the second version was decreased. Like the first prototype, the second version of the device was not designed to be a fully functioning product. It was only meant to be a test platform for the IMU portion of the motion capture device.

3.2.2 Sparkfun IMU 6DOF

After it was decided that motion capture is possible with low cost sensors, we tried a commercially available sensor made by Sparkfun Electronics [28]. The device is the IMU 6DOF v4 with bluetooth capability and was purchased from Sparkfun Electronics in 2009. It uses a Freescale MMA7260Q triaxial accelerometer ($1g$, $2g$, $4g$ or $6g$ sensitivity), two Invensense IDG500 dual-axis gyroscopes ($500\,^\circ/s$), Honeywell HMC1052L and HMC1051Z magnetic sensors, and a Roving networks RN-41 bluetooth module for wireless transmission of

the data. Figure 3.4 shows the device which was used.

While all of the specifications seemed to make this device suitable for the project, it did not perform with the accuracy necessary for motion tracking. When the IMU 6DOF was placed on a wheel spinning at a constant speed, the output produced was not linear. An accelerometer which is placed on a wheel spinning at a constant velocity should track gravity as well as the centripetal force produced by the spinning wheel. If the output of this accelerometer is plotted, it should produce a sinusoidal waveform centred around the centripetal force created by spinning motion. When the Sparkfun 6DOF sensor was placed on the wheel and its output plotted, the sinusoidal waveform was distorted. This implied that the sensor was non-linear across its output range. Non-linear output data would produce incorrect data for motion capture, resulting in inaccurate position tracking. The non-linearity of the output would normally not be a serious issue, because a linearization curve can be used to correct the data provided by the sensor. This non-linearity, however, was inconsistent across the entire output range, causing the output of the sensor to sporadically change in steps. When the sensor's data was plotted, the sinusoidal output would instantaneously change in an unpredictable manner. The amount the sensor's output changed seemed to occur randomly and no pattern could be discerned to predict when the output would change, in what direction it would change, or by how much. Even when the sensor was held stationary, the output would suddenly and unpredictably change. We were unable to determine whether the sensor firmware was responsible for the sudden output changes, or if the sensors were faulty. While tests on the device showed that the Sparkfun sensor was unsuitable for use, Sparkfun themselves acknowledged on their website that the magnetometer outputs were excessively noisy while using the device over Bluetooth. In either case, we concluded that the Sparkfun IMU 6DOF is unsuitable for motion tracking when the data must be double integrated to determine position.

3.2.3 Second Prototype

Once it was determined that the commercially available sensor was too unreliable to be suitable as a motion detector, a second prototype was designed. This prototype was designed to determine whether a low-cost IMU could be used to track movement in 3D space.

The main sensing device chosen for this project was an Analog Devices ADIS16350 [29], which is a 6DOF inertial measurement unit. At the time this device was chosen, the specifications were found to be the most appropriate for the application and the device was the most advanced 6DOF inertial measurement unit available in its price range. This sensor measures rotational velocity around three orthogonal axes and acceleration along those axes. Using the acceleration along the three axes, it should be possible to calculate the linear distance travelled by the device. Rotational velocity should give the orientation, or heading, of the device at that location in space. The ADIS16350 should provide all of the 6DOF movement data that is required as discussed in Section 3.1. Interfacing the ADIS16350 to the PC is accomplished through a microcontroller and a bluetooth transceiver. The microcontroller receives all of the sensor data and formats it for transmission to the PC. It is also used to provide a user interface that allows low-level

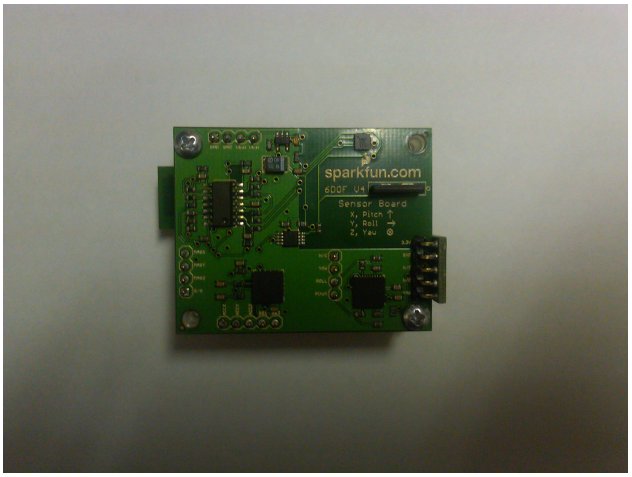

Figure 3.4: Sparkfun 6DOF IMU

sensor options (such as sample rate) to be changed from a serial port terminal window. Having the options menu programmed into the microcontroller allows the sensor board to be more easily integrated into other projects, rather than being restricted to one application. The microcontroller passes data to a bluetooth module which is configured to act as a serial port replacement. This allows the PC to see the sensor as being connected over a wired serial port, even though it is connected wirelessly through a USB port. Configuring the device this way should provide compatibility with the largest range of computer models. Figure 3.5 shows a high level block diagram of the system. Each of these blocks will be discussed in further detail in the following sections.

3.3 Sensor Board Hardware

The sensor board consists of the ADIS16350 sensor, a Microchip dsPIC30F3012 microcontroller, an RN-41 Bluetooth transceiver, a 5V power supply, a 3.3V power supply, and various switches, connectors and passive components. The 5V power supply is implemented using an LM2940T; a 1 A low dropout voltage regulator. This power supply provides power to the microcontroller and the IMU. The 3.3V voltage regulator (LM317 three terminal adjustable voltage regulator adjusted to provide 3.3V) provides power to the RN-41 Bluetooth transceiver. Appendix C shows the schematic and Appendix D shows the layout of the prototype circuit board as designed and built. The prototype board measures 2.5 inches by 4.25 inches. This is too big to be used as a final device, but the layout can be easily miniaturized using surface mount components, instead of the through-hole components used here. Through-hole components are easier to test and remove from the circuit board should there be any problems with a component. Because of this, the prototype was designed with as few surface mount components as possible. It is expected that the use of smaller, more compact components would allow the device to have a final size of two inches by two inches; a size that is acceptable for use as a wearable device.

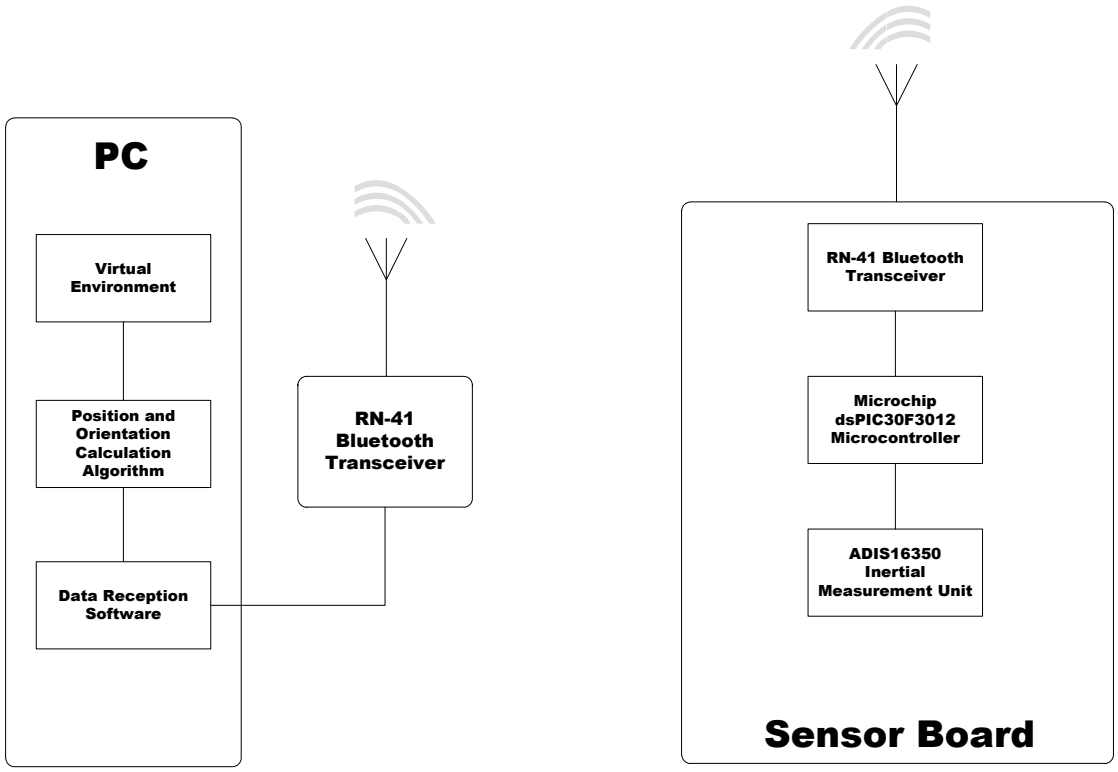

Figure 3.5: High Level System Block Diagram

3.3.1 Analog Devices ADIS16350 Inertial Measurement Unit

The ADIS16350 was chosen because at the time of hardware selection, it was the most advanced IMU sensor available in a sub-$1000 price range. It provides measurements of acceleration, rotational velocity and internal temperature, as well as providing an output indicating the status of the sensor (whether any internal errors have occurred, etc.) The temperature is given as an output because the sensor is not internally temperature calibrated, so external software temperature compensation is required. The sensor outputs digital values over a serial peripheral interface (SPI) bus. Accelerations and rotational velocities are given as 14-bit two's-complement outputs and the temperatures are 12-bit two's-complement outputs. Temperature is given as a value of zero at 25 °C and for every degree change, the output changes by 6.88LSB's. Accelerations in the range of $\pm 10g$ can be measured where one LSB of the output corresponds to $2.522mg$ and a value of zero corresponds to zero acceleration. The accelerometers have an rms output noise of $35mg$ and a temperature coefficient of 4 mg/°C. This means that for each degree of temperature variance from 25 °C, the accelerometer reading will be erroneous by $4mg$. Rotational velocities from the gyroscopes are available in one of three selectable ranges: ±300 °/s, ±150 °/s and ±75 °/s. One LSB of the gyroscope's output corresponds to 0.073 26 °/s, 0.036 63 °/s and 0.018 32 °/s for each of these ranges respectively. A gyroscope output value of zero corresponds to 0 °/s at 25 °C. The RMS noise of the gyroscope in the ±300 °/s range is 0.60 °/s when using the two-tap filter setting and the output changes by 0.1 °/s for every °C. The ADIS16350 requires 5V

for power and the typical current draw from the device in fast mode is 57 mA and 33 mA in normal mode. This power consumption is similar to other devices on the market of the same type.

3.3.2 Roving Networks RN-41

For this device to be useful, the wearer must not be hindered by wires and must be able to perform a full range of rehabilitation movements. The Roving Networks RN-41 Bluetooth module is an embedded solution which is designed to be a cable replacement device. The RN-41 can be connected to a UART-compatible port and will communicate with the port as though it was a UART device. A UART is the type of connection typically used in legacy PC serial ports. These types of connection have been an industry standard for many years and are simple to set up and use. The RN-41 is the ideal device for use as a cable replacement since most microcontrollers these days have a UART controller included as a peripheral in the chip. Since UART serial ports have been nearly phased out of PC's, USB would be the optimal solution for connection to a PC. Microcontrollers generally do not have USB controllers embedded into them and the ones that do are non-trivial to implement. This, and the fact that serial port emulators are widely available for PCs means that the UART replacement is still the best solution. An RN-41 is required for each 'cable end' which is being replaced. This means that one RN-41 device must be incorporated into the sensor device and one must be connected to the computer. With the two devices installed, all wireless communication is handled by the Bluetooth devices and communication is controlled with the standard, simple-to use UART interface.

3.3.3 Microchip dsPIC30F2011

There are almost as many microcontrollers available today as there are designs which use them. These microcontrollers come in countless configurations with various peripherals implemented in them, different power consumption characteristics, many package sizes, and numerous price points. This can make choosing a microcontroller for an application difficult. In this design, the microcontroller chosen is Microchip's dsPIC30F2011. This device offers the peripherals required to connect the IMU and the Bluetooth device, has enough program memory to store the code and the user menus, and is capable of operating at the speeds required to communicate with the sensor device and the PC. A DSP-capable microcontroller was chosen because even though the DSP functions are not presently being used to perform calculations on the incoming data (that's left for the PC to do for the time-being), it is desired that the final version of the sensor perform all of the calculations and the filtering necessary to provide stable position and orientation data without the help of the connected PC. If the sensor unit was no longer reliant on the PC for position and orientation calculations, it would be a more complete solution which could be ported to a number of other applications and it would allow the device to be connected to a slower or more compact PC or other display device such as a smartphone. While many microcontrollers may provide the functionality described here, the main reason for this device being chosen is that we are familiar with this brand and this allowed the fastest development time.

3.3.4 Power Consumption

As discussed above, the Analog Devices IMU sensor requires 57 mA of current while operating in fast mode (which is the mode the sensor will run in normally for this device). The Roving Networks RN-41 bluetooth module uses a maximum of 100 mA of current while connected (typically 30 mA) and would be connected the entire time the device is in operation. The dsPIC30F3012 microcontroller used in the project draws a maximum of 250 mA of current, though operating current of this device is much lower(typically 94 mA); most of the internal peripherals are unused and many of the pins on the device are unused. Voltage regulators LM317 and LM2940 draw maximum quiescent currents of 5 mA and 60 mA of current, respectively. Again, this is a maximum and quiescent currents will typically be lower than this (3.5 mA and 30 mA). The maximum current drawn for the device is therefore approximately $57mA + 100mA + 250mA + 5mA + 60mA = 472mA$ On average, however, the device would require $57mA + 30mA + 94mA + 3.5mA + 30mA = 214mA$. Since the device needs 5V to run, the device would require four 1.5V AA batteries in series. When batteries are wired in series, the voltage sums, but the current capacity does not. The amount of current required by the device is acceptable for use in this battery-powered sensor, as with average 1400 mA h AA batteries, this sensor should be able to run for nearly three hours and the device would only be required to be used for an hour at a time. Normal usage would allow the device to run for more than six and a half hours on a bank of four AA batteries. Testing of the device has shown that the power consumption calculated here is very modest and on the high side, as the prototype device built has been used for more than eight hours on a single set of four AA batteries.

In the final device, there is room for improvement in the power consumption and it's likely that the current consumption could be reduced to less than 100 mA with careful component choices and some power saving design techniques. A consumption of 100 mA or less would ensure that the device could be operated for an entire work day before charging would be required.

3.4 PC Implementation

Software for this project is implemented on the PC in order for the sensor's output to be used in a meaningful manner. A couple of programs were developed in order to test and confirm data correctness of the sensors. A Visual C# program was developed to connect to the sensor, accept incoming sensor data, format that data to a useful representation, and then save the data to a file. MATLAB was then used to read the sensor data from the file, filter the data, convert the data into an orientation and position representation, and save the motion data to a file.

3.4.1 Data Representation

Each measurement taken by the IMU is sent over the Bluetooth connection without any signal processing or filtering. The output of the sensor prototype was modeled after that of the Sparkfun sensor so that during testing, either of the sensors could be used to communicate with the software. Each measurement taken by the sensor is provided in a fixed-width packet with a header and footer so that the beginning and end of the packet can be tracked by the receiving computer. This type of packet provides some error-checking capabilities because the header and footer of each packet are always the same distance from one another in the data stream. A fixed-width packet also simplifies recovering data from the stream, since each piece of data is the same size. Figure 3.6 shows the general format of the data packet as it is sent over the Bluetooth channel. While sending data in binary format, this packet is a fixed length in bytes. However, when transmitting in ASCII format, the length will change based on the number of ASCII characters required.

There are two types of data representation supported by the sensor; the inertial data can be streamed as an ASCII-coded set of numbers (so that data can be watched and captured from a terminal window to verify operation) or the data can be streamed in a binary representation, which requires fewer bytes to transfer the same number of measurements. This binary stream is much faster than the ASCII representation because each measurement is represented by exactly two bytes (16 bits), whereas the ASCII stream can require up to six bytes per measurement depending on how large the number is.

3.4.2 Visual C# Program

In order to find position and orientation as measured by the IMU, the data obtained from the IMU must be streamed through the UART, parsed from the stream and then saved in a useful format. A Visual C# program was written specifically for this purpose. The program opens up the serial COM port which the sensor is connected to (in this case, it's a virtual COM port created by the driver for the RN-41 cable replacement module). It then communicates with the RN-41 cable replacement module (a Sparkfun Bluedongle-RN-USB) where it negotiates a connection to the sensor. After a connection has been established with the prototype sensor device, a command is sent to the device to tell it to start streaming data. At this point, the program begins parsing measurements out of the data stream and storing each value in memory as a string of characters representing the measurement value. The data keeps getting written to memory in this fashion until the program is stopped by the user. Once the stop button is pressed, the data from memory is written to disk as a text file with each measurement type (accel, gyro, etc.) stored in a column with each row representing a different sample. The text file includes a header on each column which describes the type

<table>
<tr><td rowspan="2">Header</td><td>Status</td><td>TempX</td><td>TempY</td><td>TempZ</td><td rowspan="2">AccelX</td><td rowspan="2">AccelY</td><td rowspan="2">AccelZ</td><td rowspan="2">Pitch</td><td rowspan="2">Roll</td><td rowspan="2">Yaw</td><td rowspan="2">Footer</td></tr>
<tr><td>Sample</td><td>MagX</td><td>MagY</td><td>MagZ</td></tr>
</table>

Figure 3.6: General data packet format for transmission over the bluetooth channel

of data in that column. After the file is written, the program sends a command to the sensor to stop the data transfer and the program closes the COM port connection. A block diagram of the Visual C# program is given in Figure 3.7.

3.4.3 MATLAB Algorithm

After the data has been transmitted and saved to the computer, a MATLAB algorithm converts the raw data into orientation and position data. The data is stored in arrays of samples for each of the six IMU measurements (accelX, accelY, accelZ, pitch, roll, yaw). Pitch is the rotational velocity around the x-axis, roll is about the y-axis, and yaw is about the z-axis.

The gyroscopes in the IMU have a sensitivity of $0.073\,26\,^\circ$/s/LSB and the range of the gyroscopes are $\pm 300\,^\circ$/s so the range of rate values which the sensor provides is $\pm 300\,^\circ$/s / $0.073\,26\,^\circ$/s/LSB = ± 4095. If the IMU is rotated at $\pm 100\,^\circ$/s directly about the z-axis (by rotating the x-axis toward the y-axis), then the pitch and roll outputs will be zero and the yaw output will be 1365. An image showing the rotation described here is displayed in Figure 3.8.

The accelerometers have a range of ± 10 g and a sensitivity of 2.522 mg/LSB, so the range of values generated by the accelerometers is ± 3965. With a sensitivity of 2.522 mg/LSB, a measurement of 1 g (which is the measurement of gravity) would produce a decimal value of 397. If the axes of the IMU are positioned with two axes perpendicular to gravity and one axis parallel to gravity (pointing away from the centre of the earth), the perpendicular axes will produce values of zero and the parallel axis will produce a value of -397. The orientation of the axes in this position is shown in Figure 3.9.

The decimal values provided by the sensor need to be converted to more useful formats and the data given by the sensors have some errors and bias which need to be removed before the data can be used. The first thing which needs to be found is the offset of the gyros. The gyros should give a value of zero when the device is stationary. A nonzero output from the gyroscope when stationary is called the offset. The offset of each axis is subtracted from the values provided by the gyroscope. With the offset removed, the rotational velocity of each axis is found by multiplying the offset-free samples by the gyroscope scale ($0.073\,26\,^\circ$/s/LSB). This gives the rotational velocity in $^\circ/s$. To find the angle which each axis has rotated through during each sample period, the rotational velocity is multiplied by the sample period.

In order for the angle to be used to find the current orientation, a rotation of the previous orientation must be done using the current rotation angles. There are a few different methods for rotating the previous orientation to the current orientation. Axis-angle representation is one option. In an axis-angle rotation, the current coordinates are rotated around a given axis through a given angle. This method is not optimal for the project, because the data we have includes three angles and no axis coordinates. Issues with axis-angle rotations include the fact that a rotation of zero degrees is not uniquely defined and consecutive rotations do not follow vector math, making consecutive rotations nontrivial.

Euler angles are a set of three angles which represent rotations about each of the three axes. Rather

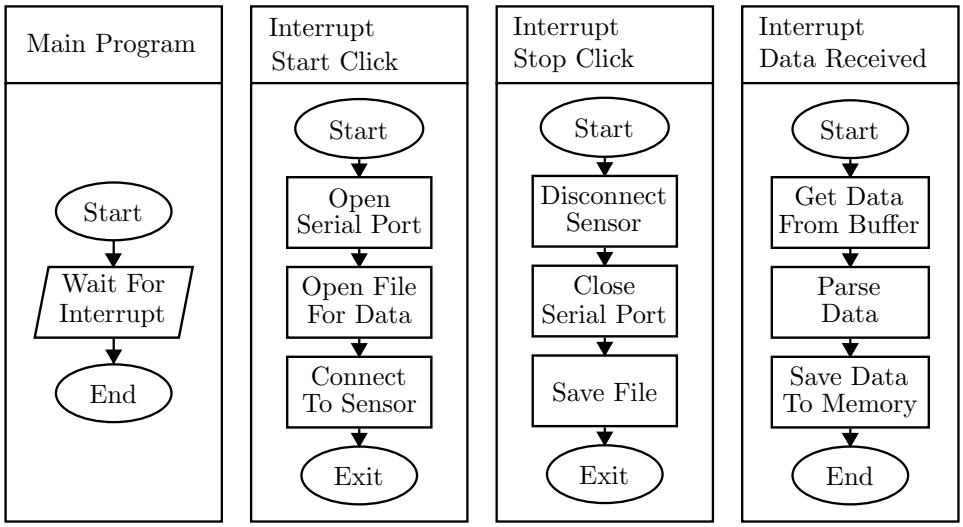

Figure 3.7: Visual C# program flowchart

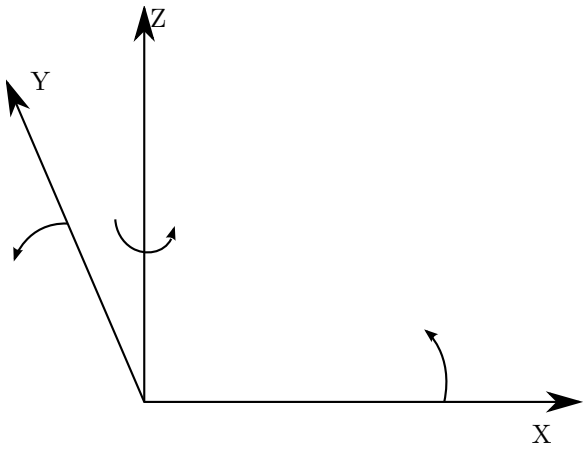

Figure 3.8: Rotation about the z-axis

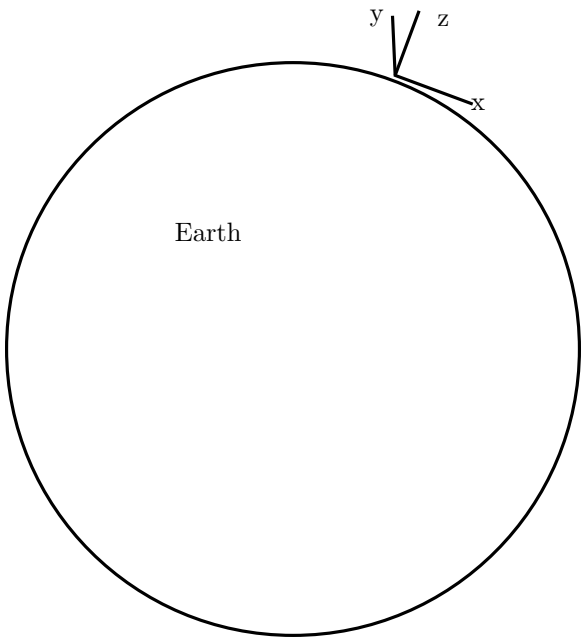

Figure 3.9: Position of IMU when measuring gravity with the Z axis only

than being a rotation through three angles, the Euler angle rotation is three rotations, each around a single axis. This type of rotation is useful if the rotations happen around one axis at a time but is less useful for angles which are rotated through simultaneously, as in the measurements that the gyroscopes provide. As well as not being useful for simultaneous rotations, Euler angles suffer from singularities when the rotation angle is at or around 90°. It is obvious that movements made by an arm will be performed through a range of angles larger than ninety degrees, since moving the arm from hanging at a person's side to reaching the arm straight out in front of the body is a movement of 90°.

A third way of representing rotations in three-dimensional space is to use quaternions. Quaternions overcome some of the issues arising from the order-of-rotation operations that will be present if Euler angles are used and quaternions are immune to singularities due to gimbal lock, another Euler angle issue. For these reasons, quaternions are used in this paper in order to track rotation and orientation information. Section 3.4.3 discusses how quaternions work and how they are used to track orientation.

Quaternion Representation of Orientation Data

As was discussed in Section 2.3.3, quaternions are a way to represent three-dimensional quantities in four dimensions. Coutsias and Romero [30] show that a rotation of a vector by quaternions is accomplished by

$$n_r = q n_i q^{-1} \tag{3.3}$$

where n_i is the vector in its initial position and n_r is the vector after it has been rotated. The vector n_i, a three-dimensional quantity, must be given as a four-dimensional quaternion. This is as simple as setting the real part of the quaternion to zero and setting the x, y, and z parts of the quaternion to the x , y ,

and z values of the three-dimensional vector. The three-dimensional rotated vector is extracted from the rotated n_r in a similar manner, by using the x , y , and z parts of n_r as the x , y , and z coordinates in three-dimensional space, respectively.

Another way of looking at these rotations is to picture them as a transformation from one coordinate system to another, when the quaternion required to transform from one to the other is known. This way of imagining the rotation is more in line with the situation in this project. One coordinate system is oriented with the earth and the other coordinate system is oriented with the sensor body. As the sensor rotates, there exists a rotation which will convert measurements made in one coordinate system to the other. A quaternion can be used to represent this rotation. As long as we continually keep track of the quaternion which describes this rotation, the orientation of the sensor can be tracked in the world coordinate system.

On his website, `www.euclideanspace.com`, Martin Baker provides an excellent explanation of how Euler angles and quaternions are related and how to convert from one to the other. He provides the transformation as given in Equation (3.4). The transformation transforms from Euler angles to quaternions, with the benefit that the order of the angles put into the equation is unimportant. In this equation, θ is the roll angle, ϕ is the yaw angle, and ψ is the pitch angle.

$$\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\theta/2)\cos(\phi/2)\cos(\psi/2) - \sin(\theta/2)\sin(\phi/2)\sin(\psi/2) \\ \sin(\theta/2)\sin(\phi/2)\cos(\psi/2) + \cos(\theta/2)\cos(\phi/2)\sin(\psi/2) \\ \sin(\theta/2)\cos(\phi/2)\cos(\psi/2) + \cos(\theta/2)\sin(\phi/2)\sin(\psi/2) \\ \cos(\theta/2)\sin(\phi/2)\cos(\psi/2) - \sin(\theta/2)\cos(\phi/2)\sin(\psi/2) \end{bmatrix} \tag{3.4}$$

The orientation of the sensor is found by tracking the sensor with respect to a starting location. Since the accelerometers are always measuring gravity, we want to keep track of where gravity is in the current frame. Using the orientation calculated from the gyroscope measurements, we can track the location of gravity. This way we will know which accelerations are due to movement and which are due to gravity. With each measurement sample, the new orientation is found by multiplying the previous measurement by the new quaternion. As stated in Section 2.3.3, the rotation from sensor body measurements to world coordinate measurements is given by

$$n_e = q_{be} n_b q_{be}^{-1} \tag{3.5}$$

where n_b is the vector in the sensor body coordinates, n_e is the vector in earth coordinates, and q_{be} is the quaternion that transforms from sensor body to earth coordinates. The thing that makes quaternions so perfect for these consecutive rotations, is that successive rotations can be accomplished simply by multiplying quaternions successively. If the device is rotated by q_1 first and q_2 next, the rotation is accomplished by

$$n_e = q_2 q_1 n_b q_1^{-1} q_2^{-1}. \tag{3.6}$$

In order to measure movement accelerations, the value of gravity must be subtracted from the measurement. During the initial calibration time (when the offset of the gyroscopes is found), the initial gravity

vector is found by averaging the measurements given by the accelerometers. This vector is multiplied by the quaternions and the resultant value is subtracted from the initial gravity vector. The previous vector is multiplied by the quaternion for the current sample, propagating the orientation through time. This is all accomplished in MATLAB.

Chapter 4

Experimental Setup and Results

In this chapter, the experimental procedures used for testing the linear displacement sensor and the inertial measurement device are described. The results of the tests are provided following the descriptions of the test methods.

4.1 Linear Displacement Sensor

A linear displacement sensor was attached to a prosthetic arm as shown in Figure 4.1. The documentation for the linear displacement sensor [31] gives the stroke length as 109.3 mm and the output as 0.1 V to 4.9 V ratiometric with 5 V. Upon testing the sensor, it was found that the stroke length was closer to 100mm. The range of digital values produced by the sensor within the stroke length was 11 to 3588, when using a 12-bit ADC. Above 3588, the output of the sensor was non-linear, as the values given by the sensor did not change proportionally with the amount of movement applied to it. In the lower "dead zone" of the sensor, the output gave values of 10 and 11, no matter where in the dead zone the sensor was reading. Between values of 11 and 3588 (corresponding to 0 mm and 99.4 mm, respectively), the output appeared to be linear. This can be seen in Figure 4.2 where the sensor was tested while being moved 1 mm at a time across the output range of the sensor. In order to get the 1 mm increments, a ruler was placed next to the sensor and the sensor was moved based on 1 mm on the ruler as closely as possible. Obviously, there is a large amount of human error present in this measurement, but it is reasonable because it is only used to demonstrate that a 1 mm movement gives approximately the same change in value across the operating range of the sensor. A small amount of nonlinearity in this sensor would not be a problem since the value given will be directly used as a measurement. If the sensor is out by even 2mm (approximately 70 LSB's) for any given distance measurement, the error will likely be imperceptible by the user. The timesteps between movements were kept as close to constant as possible, so that the graph would show a relatively constant slope as time went on. Figure 4.3 shows a portion of Figure 4.2 zoomed in to demonstrate the step size when a movement of approximately 1 mm is made.

From the data in Figure 4.2, the average step size can be found. Using MATLAB, the average size between each step was found. The size of each step is given in Figure 4.4. The horizontal line shows the median step size, 35.5 LSB. This tells us that a nominal 1 mm step size made across the sensor's useful range

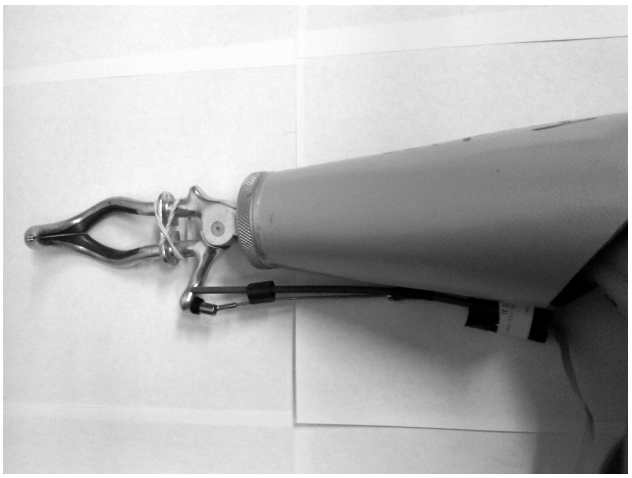

Figure 4.1: Prosthetic arm with sensor attached to the cable.

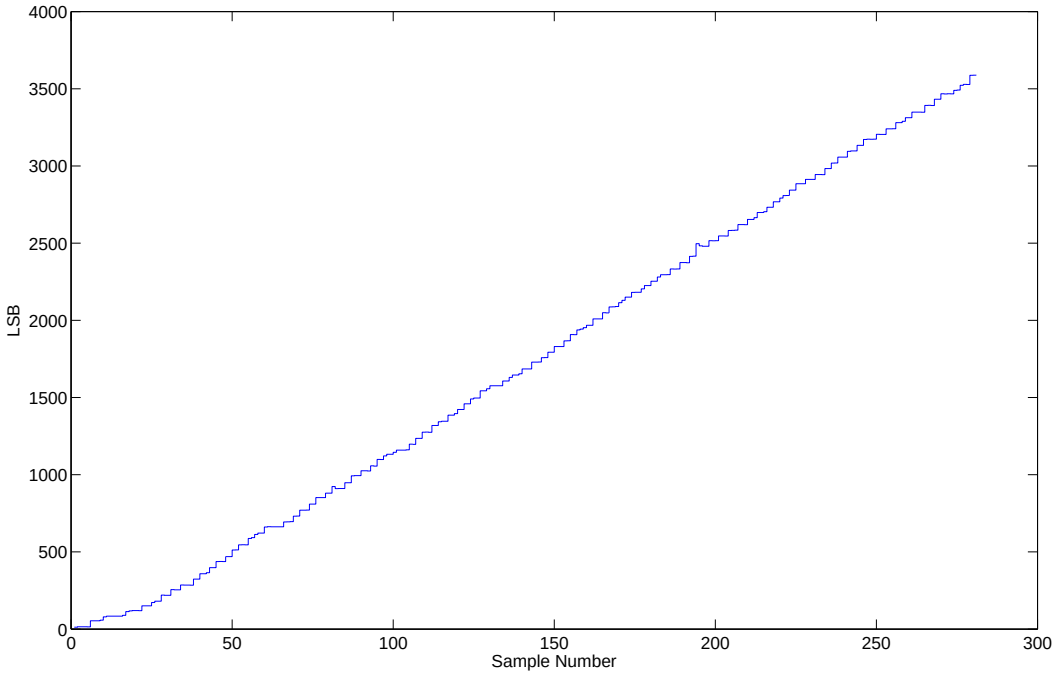

Figure 4.2: Linear displacement sensor output when moved in 1mm increments

will change the output by 35.5 LSB.

Experimentally, we have calculated that the sensitivity of the linear displacement sensor is 35.5 LSB/mm, or 0.028 17 mm/LSB. The sensor was found to output a value of 11 when at a minimum (even though according to the manufacturer, it shouldn't have put out any value below 82: $0.1V/5V * 4095 = 82$). The largest reading that the sensor gave was 3588, at a distance of 99.4 mm. This gives a resolution of

$$resolution = \frac{3588\,\text{LSB} - 11\,\text{LSB}}{99.4\,\text{mm}} = 36\,\text{LSB/mm}.$$

In inches, this is 914 LSB/in.

The resolution of the sensor was verified by moving the sensor three inches within its useful range. Figure 4.5 shows the movements made and the distance calculated. The output clearly shows that the sensor has been moved three inches, from 0.5 inches to 3.5 inches.

Since the prosthetic's hook doesn't actually open three inches when the cable has been pulled three inches, it's necessary to use a transfer function to relate the distance the cable is pulled to the distance that the hook has opened. The transfer function in this case is straightforward since the end of the hook is rigidly attached to the point where the cable attaches and both the cable attachment and the end of the hook use the same pivot point. The arc that the cable attachment point moves along provides the angle that the hook opens through by the equation

$$\theta = \frac{a_{cable}}{r_{cable}},$$

where a is the arc length that the cable attachment point travels through (which is equal to the distance that the cable has been pulled) and r is the distance between the cable attachment point and the pivot point of the hook. The arc that the hook end travels along is given by

$$\theta = \frac{a_{hook}}{r_{hook}}.$$

Since the angle that the hook travels through is the same as the angle that the cable attachment travels through, these equations can be equated:

$$\frac{a_{hook}}{r_{hook}} = \frac{a_{cable}}{r_{cable}}.$$

The distance from the cable connection to the pivot point of the hook is 4.5cm (r_{cable}). The distance from the end of the hook to the pivot point is 10cm (r_{hook}). Substituting for r_{hook} and r_{cable} leaves us with Equation 4.1.

$$a_{hook} = 2.22 * a_{cable} \tag{4.1}$$

Figures 4.6 and 4.7 show the values given when the hook is opened in four steps and then closed in four steps. The hook went from closed to open approximately four inches, half an inch of the cable being pulled each time. It was then closed in 0.5 inch cable steps. Figure 4.6 shows the distance that the cable has been pulled as measured and Figure 4.7 shows the distance that the hook has been opened based on Equation (4.1).

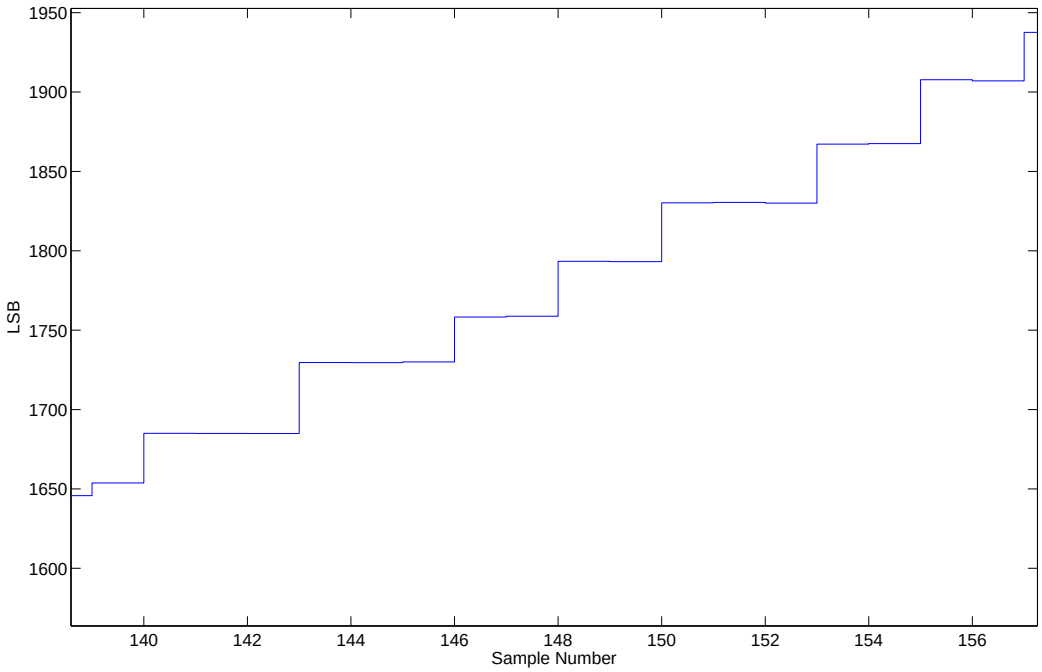

Figure 4.3: Linear displacement sensor output when moved in 1mm increments (zoomed in)

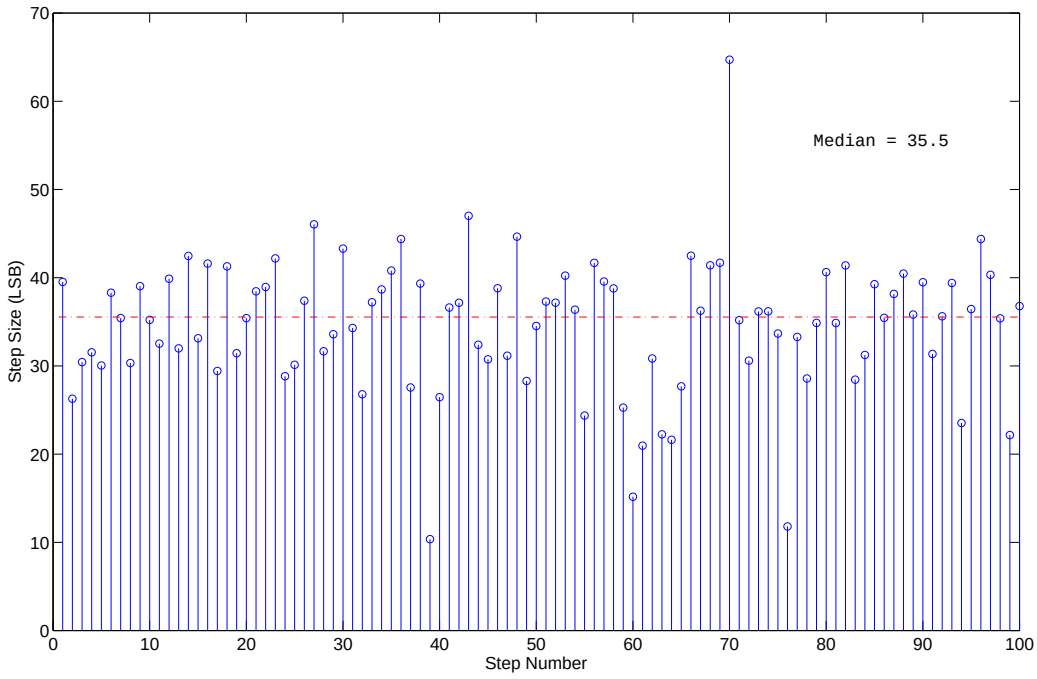

Figure 4.4: Size of each 1mm step. Horizontal line is the median.

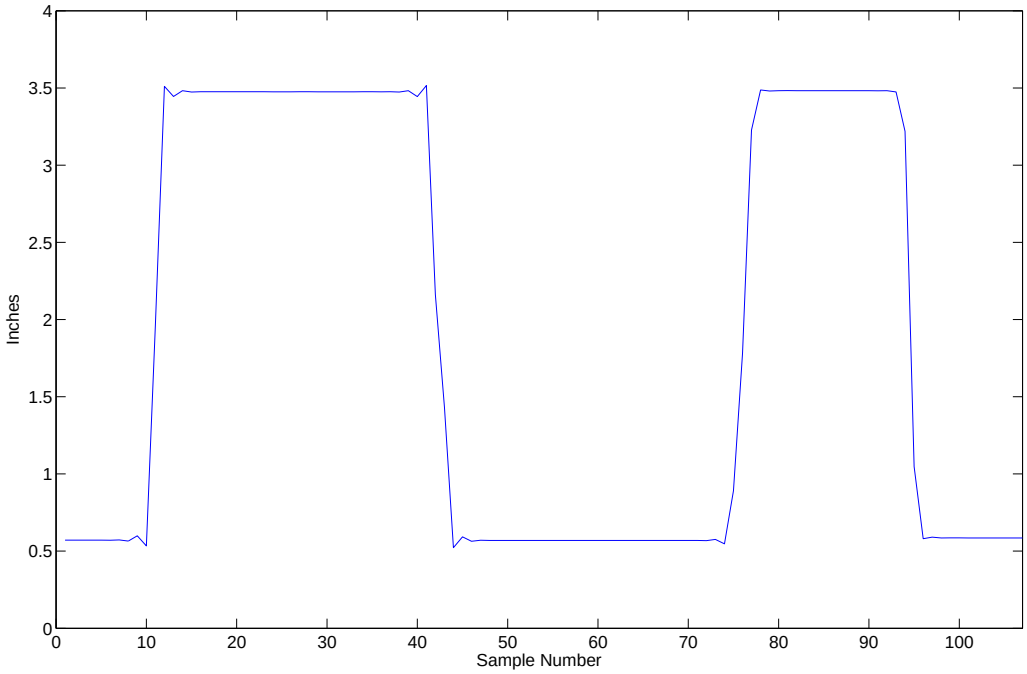

Figure 4.5: Sensor magnet moved three inches along the waveguide and back twice.

The sensor device attached to the prosthetic was used to perform a test to ensure the validity of the transfer function given in Equation (4.1) and to ensure that the resolution of the sensor found using the other methods was correct. The starting position of the magnet on the waveguide when the hook was fully closed provided an output of 3099 from the prototype's microcontroller. The position of the magnet was marked on the waveguide as a reference point. This point corresponds to $0mm$ on both the cable and the hook. The cable was then pulled and clamped into place at a random position. When the cable was clamped into place, the output of the prototype microcontroller changed to a value of 2038. This is a difference of $3099 - 2038 = 1061$. Using the resolution of 35.5 LSB/mm found earlier, 1061 LSB corresponds to 29.89 mm. The distance that the cable had moved as measured using a caliper was 29.85 mm. The percentage difference between the measured and the calculated distances is:

$$\frac{29.89 - 29.85}{\frac{29.89+29.85}{2}} = 0.134\%.$$

This result shows that the value of 35.5 LSB/mm found earlier is accurate. Table 4.1 shows the results when a few different cable pull lengths were tested.

While the table shows that the calculated cable pull distance is fairly accurate (less than 5% difference between calculated and measured, and an average difference of 2.40%), the zero point continually shifts between movements. This is likely due to the fact that the prototype is held together with tape and the magnet running along the linear displacement sensor's waveguide is hardly held in place rigidly. To see if

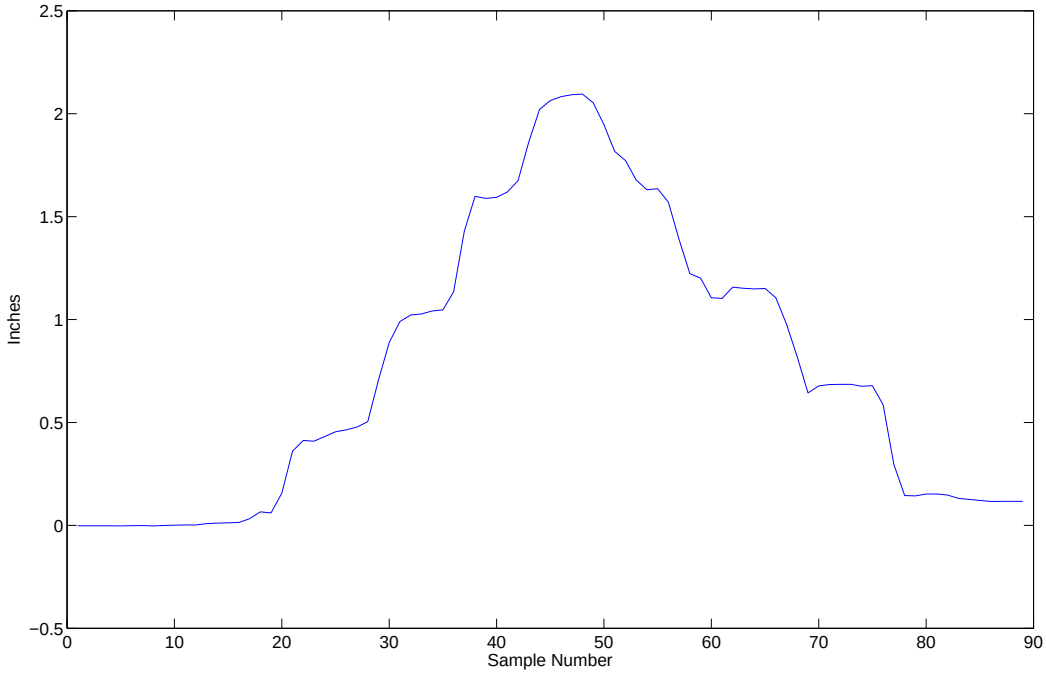

Figure 4.6: Measurement of the distance the cable is pulled at 0.5 inches at a time.

this would be an issue in practice, an average zero point was assumed, as that is likely to be the case when in operation. Table 4.2 shows the same measurements had we taken them assuming a constant zero point. By averaging the zero point, the results actually improve (which is expected since we are effectively low-pass filtering the zero point). When an average zero point of 3074 is used, the worst-case scenario difference becomes 3.66% and the average difference becomes 2.11%.

With the cable pulled 29.85 mm, the hook was measured to be open 65.75 mm. Using Equation (4.1) and the average-zero calculated cable pull distance (29.18 mm), the expected hook opening is 64.78 mm. The percentage difference between the measured and calculated hook openings is

$$\frac{65.75 - 64.78}{\frac{65.75+64.78}{2}} = 1.49\%.$$

Table 4.3 contains the results of a few similar tests at different hook opening distances. The difference between the measured and calculated hook openings are all under 10%, which we estimate be sufficient for use. As expected, the error in calculated opening values are larger than those of the cable pull lengths due to the errors from the cable calculation being compounded in the transfer function. Another likely source of error in the hook opening values is the actual measurement taken on the hook as it was opened, which is more difficult to accurately measure than the magnet on the waveguide.

The results of these tests show that the sensor chosen is accurate enough to model the hook's movement in a virtual environment. A sensitivity of 0.028 17 mm/LSB is certainly sufficient as in a virtual world it is

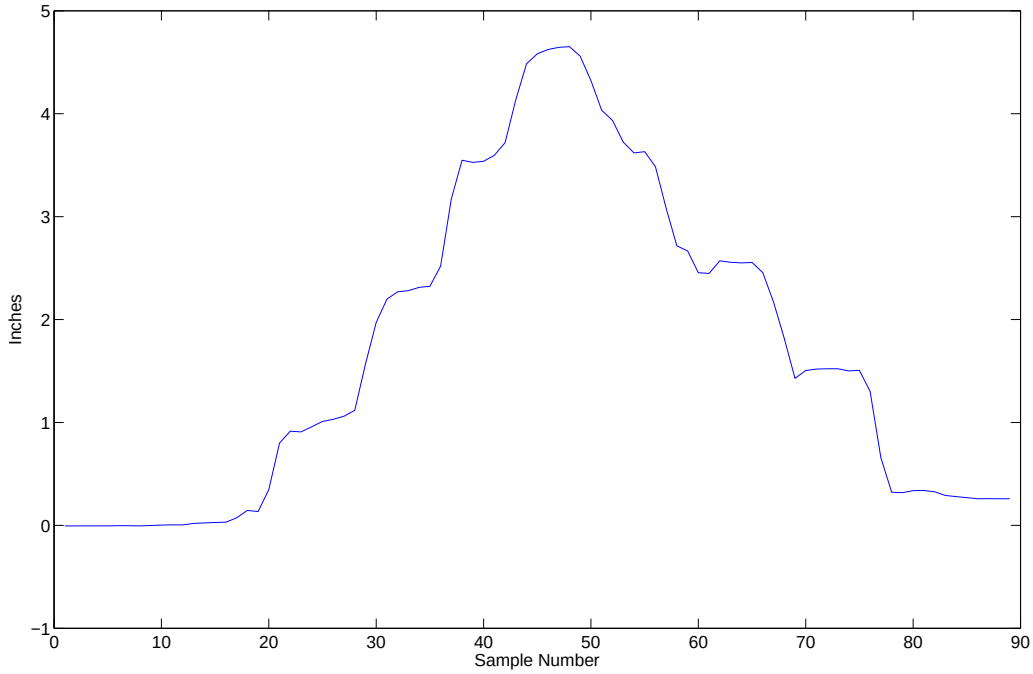

Figure 4.7: Measurement of the distance the hook has opened when the cable is pulled 0.5 inches at a time.

unlikely that movements of less that 2mm would be noticeable by a user.

4.2 Device Error

Motion tracking involves following an object's position in space and determining what direction an object is facing, otherwise known as its orientation. If the object was to always face the same direction but move in any direction up or down, left or right, or forward and backward, then three accelerometers are needed. If the direction that the object is facing can change in any direction, then three gyroscopes are needed as well as the three accelerometers needed to track movement.

Each of the accelerometers and gyroscopes will have some amount of error associated with each measurement. These errors must be taken into account before the data can be used in any motion tracking algorithm. In his technical report, "An introduction to inertial navigation," [32], Oliver J. Woodman describes some of the errors exhibited by these devices and provides some insight as to how to find and correct those errors. Over the next few sections, we will use the information provided by Woodman to analyze some of the errors present in the prototype, show their effects on the results, and implement some methods to correct those errors.

Table 4.1: Cable distance: measured vs. calculated.

Zero Reading (LSB)	Open Reading (LSB)	Difference (LSB)	Calculated Opening (mm)	Measured Opening (mm)	Difference (%)
3073	2636	473	12.31	12.8	3.90
3045	2311	734	20.68	21.35	3.19
3099	2038	1061	29.89	29.85	0.134
3080	1904	1176	33.13	32.35	2.38

Table 4.2: Cable distance using average zero point: measured vs. calculated.

Zero Reading (LSB)	Open Reading (LSB)	Difference (LSB)	Calculated Opening (mm)	Measured Opening (mm)	Difference (%)
3074	2636	438	12.34	12.80	3.66
3074	2311	763	21.49	21.35	0.654
3074	2038	1036	29.18	29.85	2.27
3074	1904	1170	32.96	32.35	1.87

4.2.1 Accelerometer Errors

The accelerometer errors which Woodman [32] discusses include constant bias error, velocity random walk, bias stability, temperature effects, and calibration errors. He says that the main causes of error in a system such as this are bias errors and random walk (white noise) errors. Through testing, it was shown that temperature effects are also very prevalent in this system. Each of these three errors will be discussed in their own sections. The first error which can be measured is the effect of temperature on the accelerometers.

Accelerometer Temperature Coefficient

The offset (or bias) of this device drifts with temperature. This is common amongst IMU devices and must be compensated for in software. To test the temperature dependency, the IMU was placed on a flat surface and held stationary for the duration of the test. The sensor was then cooled and heated using a 'cold shot' compressed air canister and a heat gun. A total of $268,196$ samples were taken over 46 minutes, 22 seconds. Figures 4.8, 4.9 and 4.10 show the temperature drift of each accelerometer axis.

As can be seen from the accelerometer temperature drift plots, the temperature in the accelerometer changes linearly with temperature along each axis. It is obvious from these plots that the temperature-related bias drift is the main source of noise in this dataset. A best-fit line has also been placed on each of the plots and the correlation coefficient between the sensor data and the best-fit line is shown in the legend next to the equation for the line. In each case, the correlation coefficient is greater than 0.98. This means

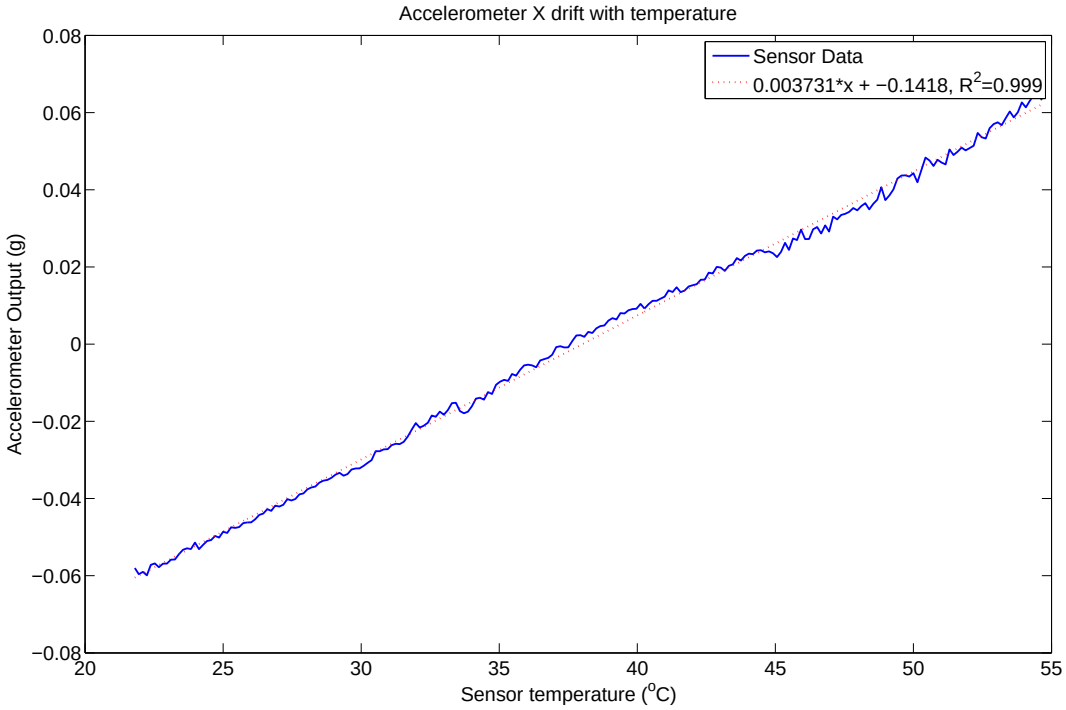

Figure 4.8: Temperature drift of the accelerometer's x-axis

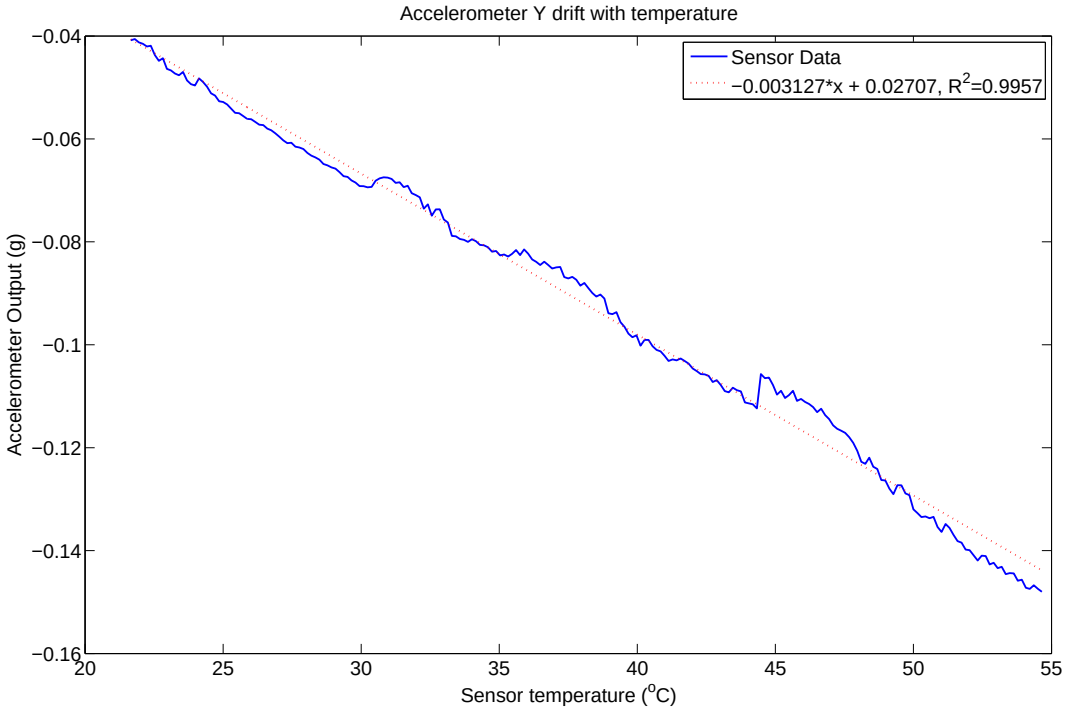

Figure 4.9: Temperature drift of the accelerometer's y-axis

Table 4.3: Hook opening using average zero point: measured vs. calculated.

Cable Pull Calculated (mm)	Hook Opening Calculated (mm)	Hook Opening Measured (mm)	Difference (%)
12.34	27.39	25.60	6.76
21.49	47.71	49.80	4.29
29.18	64.78	65.75	1.49
32.96	73.17	76.05	3.86

that the best-fit lines chosen are very good approximations for the change in output with a given change in temperature. The documentation for the ADIS16350 [29] states that the accelerometer bias drift with temperature should be 4 mg/°C. This temperature coefficient can be directly read off the plots as the slope of the best-fit line. The x-axis of the accelerometer has a temperature coefficient of 3.73 mg/°C, the y-axis has a temperature coefficient of −3.13 mg/°C and the z-axis coefficient is −5.98 mg/°C. The datasheet value of 4 mg/°C is reasonably close, but there is no way to know whether the axis has a positive or negative coefficient without testing. Temperature coefficients found in this test (Table 4.4) will be used throughout the remainder of the design to normalize the values to a temperature of 25 °C. Figures 4.11, 4.12 and 4.13 show the raw data which was used to find the temperature coefficients along with the same data once normalized to 25 °C.

The temperature compensated data shows that using a linear estimation for temperature dependency is valid as long as the temperature doesn't change too quickly. The graphs show that with slow temperature change, the output changes linearly with temperature. When the temperature of the sensor changes rapidly, the change in measured acceleration becomes nonlinear. This shouldn't cause a problem for the device, as it is unlikely that the sensor will be subjected to a rapid change in operating temperature upon normal use as a motion capture device. Any temperature change that the device sees will be caused by internal heating and from being close to the skin. Neither of these heat sources will result in rapid temperature changes.

Table 4.4: Actual temperature coefficient of the accelerometers.

Axis	Temperature Coefficient ($mg/^oC$)
x	3.73
y	-3.13
z	-5.98

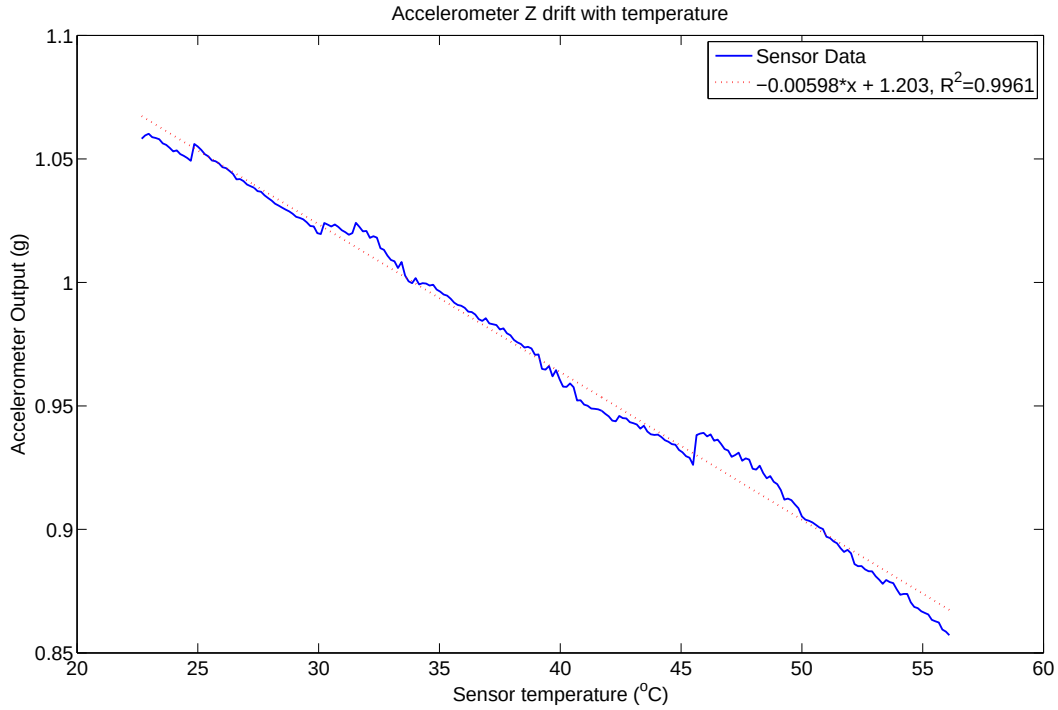

Figure 4.10: Temperature drift of the accelerometer's z-axis

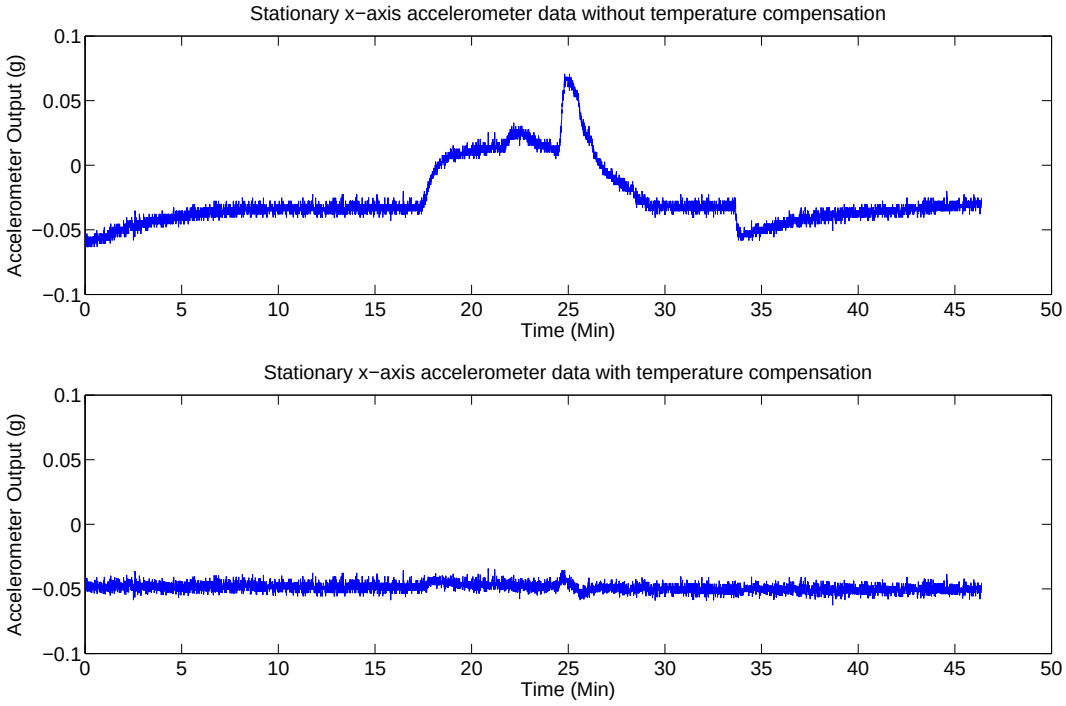

Figure 4.11: Stationary acceleration along the x-axis

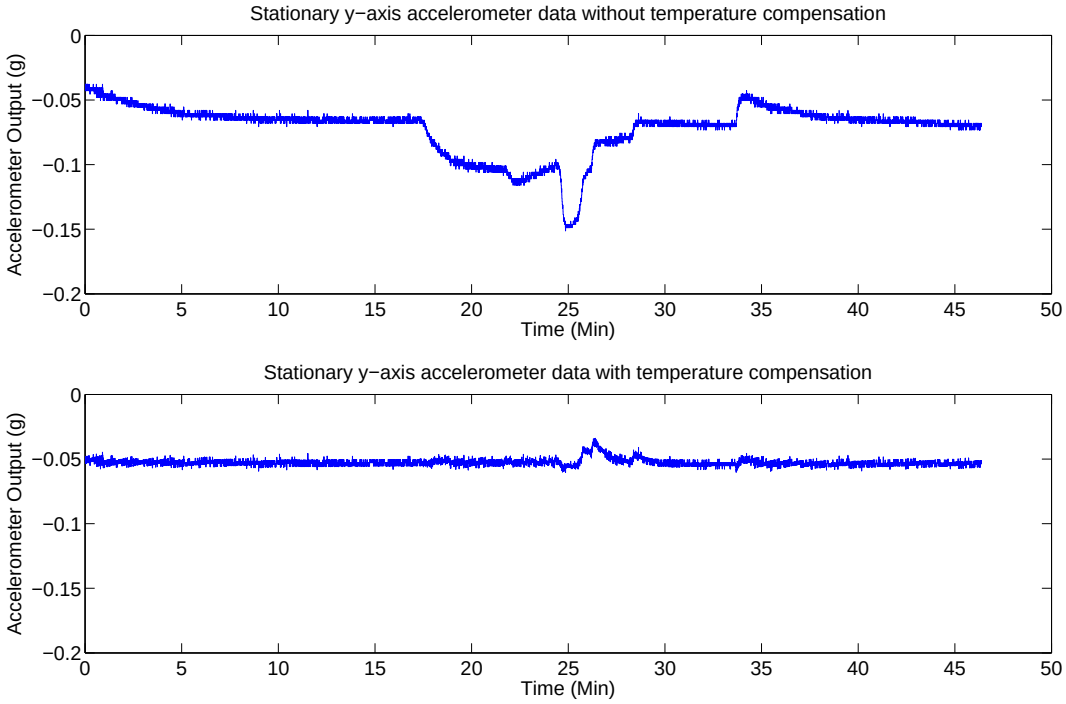

Figure 4.12: Stationary acceleration along the y-axis

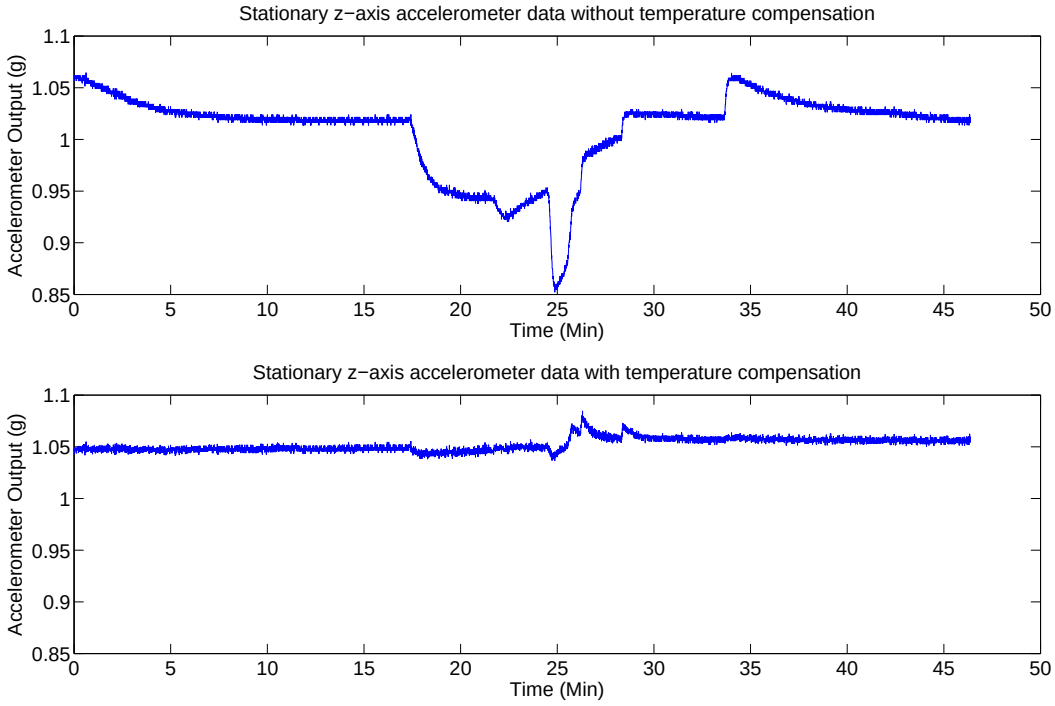

Figure 4.13: Stationary acceleration along the z-axis

Accelerometer Offset

Accelerometer offset (or constant bias) is another one of the calibration errors which may be present in an IMU. Using the IMU prototype developed for this project, the gravity vector was measured using a single accelerometer. The accelerometer gives a value of 394 when set on a table and oriented in the direction of gravity. According to the datasheet for the ADIS16350 [29], we should expect a value of 397, which means that our table is not perfectly level with respect to gravity or there is a constant bias on the accelerometer, or both. It's not important that the table be perfectly level with gravity, as much as the value read should be the same but negative when the device is turned upside down on the table.

In order to see if the accelerometer behaves as expected, the offset was first tested. When the sensor is placed on a flat surface and then flipped upside-down, the two readings should average out to zero - this will apply no matter what orientation the sensor is in with respect to gravity, provided it is flipped exactly 180 degrees. If there is any offset in the accelerometer reading, it manifests in the amount that average of the two measurements differ from zero. By averaging a large number of samples in each orientation, the average value that the IMU provided facing in one direction was 397.87 and the average value in the other direction was -393.18. The average value of these two numbers is 2.345. This is the amount by which the accelerometer's reading is offset. In order to compensate for these errors, the IMU module has a function which allows an offset to be stored in the module itself and applied to each measurement, though the offset can also be applied in software; something that would be necessary in the case that the offset drifts with time or temperature, or if the offset changes between power cycles. There are three accelerometers in this IMU and this method of calculating the offset was applied to all three. The results are provided in Table 4.5. The resulting offset of each of the axes was then used in software to correct the zero-bias of the accelerometer.

Accelerometer Velocity Random Walk

Random walk noise, also known as white noise, usually contributes the most to the noise properties of a MEMS device. Since it is random, it is also amongst the most difficult to remove from a signal. The Allan Variance technique, as shown by Woodman [32], can be used on the accelerometers' outputs to find many noise characteristics of the sensor. For the purpose of this project, it is most useful for finding the velocity random walk of the sensor.

Table 4.5: Offset calculation for each accelerometer axis.

Axis	Value Before Flip	Value After Flip	Average (Offset)
x	-10.06	-11.67	-10.86
y	-30.08	-6.93	-18.51
z	397.87	-393.18	2.35

Woodman [32] describes the method for calculating the Allan Variance:

1. Take a long sequence of data and divide it into bins of length t. There must be enough data for at least 9 bins (otherwise the results obtained begin to lose their significance).
2. Average the data in each bin to obtain a list of averages $(a(t)_1, a(t)_2, \cdots, a(t)_n)$, where n is the number of bins.
3. The Allan Variance is then given by

$$\text{AVAR}(t) = \frac{1}{2 \cdot (n-1)} \sum_i (a(t)_{i+1} - a(t)_i)^2$$

To determine the characteristics of the underlying noise process, Allan Deviation

$$AD(t) = \sqrt{\text{AVAR}(t)}$$

is plotted as a function of t on a log-log scale.

The data from the accelerometers were temperature compensated before the Allan Deviation plot was created. The sensor was left stationary for 45 hours. This resulted in approximately 8.1 million samples taken for each axis. Two thousand Allan Variance measurements were done, with bin lengths logarithmically spaced between $t = 40.28ms$ and $t = 33.57s$ This results in 4 million bins at the shortest averaging time and 81 bins at the longest averaging time, which satisfies the minimum of 9 bins criteria set forth by Woodman.

The Allan Deviation plots for the accelerometers are given in Figures 4.14, 4.15, and 4.16. According to Woodman, the random walk component appears as a sloped portion of the graph with a gradient of -0.5. He says that the random walk component can be read directly from the graph by fitting a straight line through that slope and reading the value of the line at an averaging time of $t = 1$. The results of the random walk measurements for the accelerometers are shown in Table 4.6. These results seem reasonable, due to the fact that Woodman tested similar MEMS sensors and got similar results, though they seem to be much lower than the specified 2 $m/s^2/\sqrt{h}$.

As will be seen when the acceleration is integrated to find position, all of the above discussed error sources contribute significantly to the final result.

4.2.2 Gyroscope Errors

Since the gyroscopes and accelerometers used in this device are MEMS sensors, the error sources which each possess are similar. The gyroscopes are affected by temperature effects, though not as significantly as

Table 4.6: Velocity random walk for each accelerometer.

Accelerometer Axis	Velocity Random Walk Coefficient $(m/s^2/\sqrt{s})$	Velocity Random Walk Coefficient $(m/s^2/\sqrt{h})$
x	0.0028	0.168
y	0.0016	0.096
z	0.0017	0.102

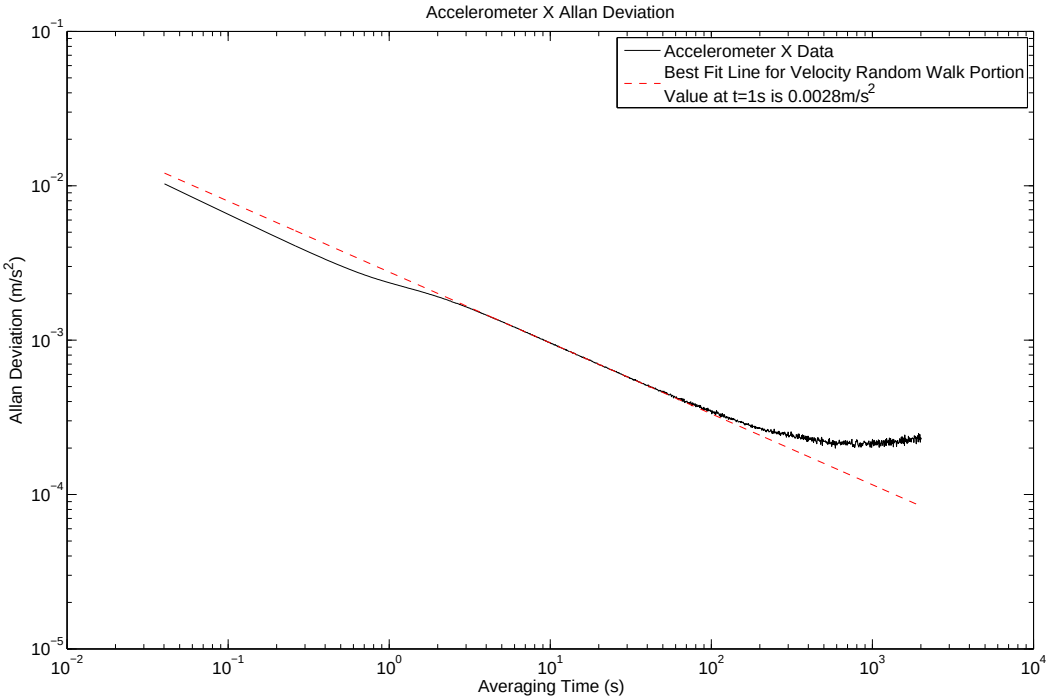

Figure 4.14: Allan Deviation plot of the x-axis accelerometer

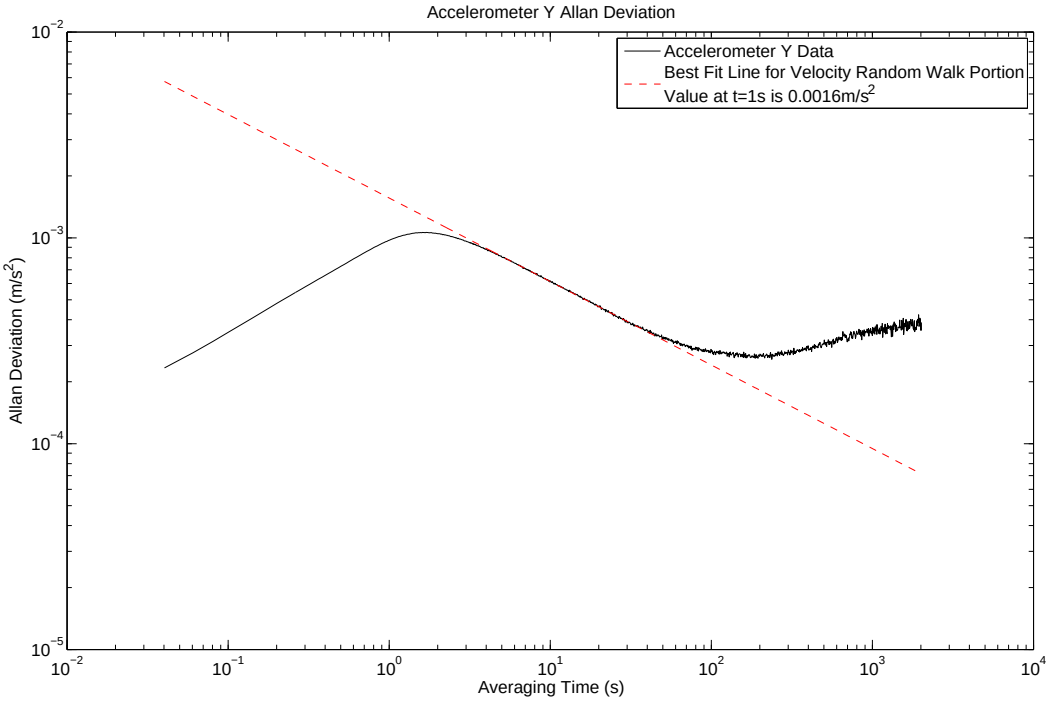

Figure 4.15: Allan Deviation plot of the y-axis accelerometer

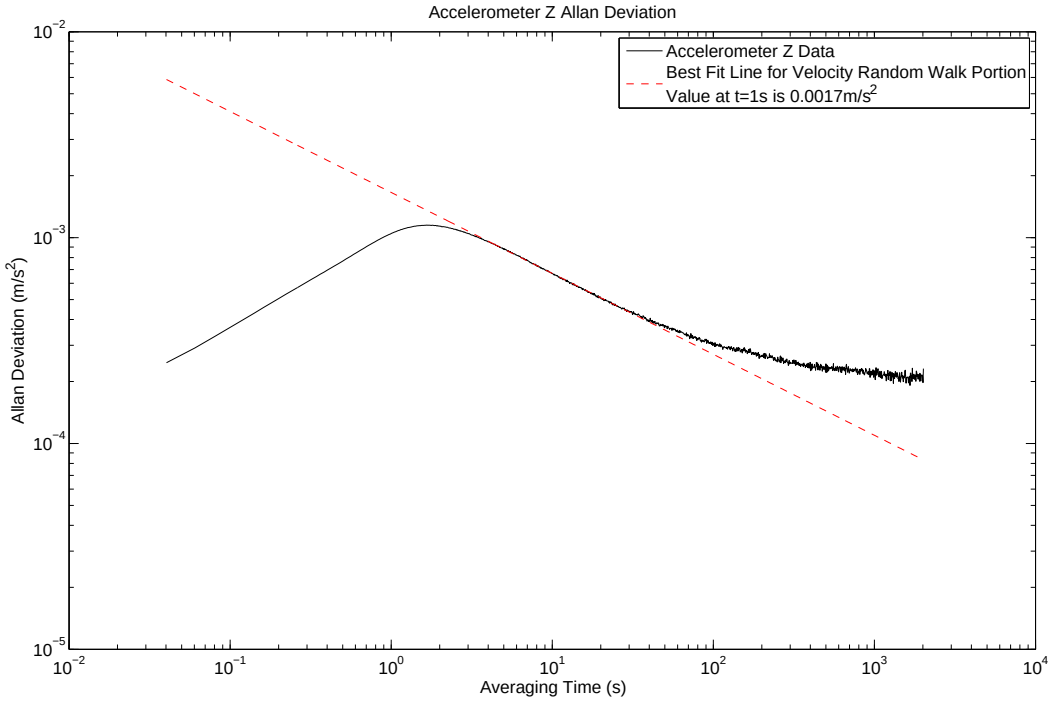

Figure 4.16: Allan Deviation plot of the z-axis accelerometer

the accelerometers. As well, the gyroscopes will have a constant bias which must be removed before the angular velocity measurements are integrated. MEMS gyroscopes are all affected by random walk just as the accelerometers are, but since gyroscope signals are only integrated once to give an orientation measurements, the result of that error in the final answer is not as significant as in the case of the accelerometers.

Gyroscope Temperature Coefficient

In the same manner that the accelerometers' biases change with temperature, so do the gyroscopes'. By using the same method for finding the temperature coefficients of the accelerometers, the temperature coefficients of the gyroscopes were found. The results of the temperature coefficient tests are given in Figures 4.17, 4.18, and 4.19.

While the y-axis and z-axis gyroscope have fairly linear temperature drift curves (a change in temperature results in a similar change in output), the x-axis gyroscope shows a highly nonlinear temperature. While this makes it more difficult to correct for temperature drift in that axis, the temperature drift in the x-axis is nowhere near as severe as in the other axes. Where the y- and z- axes have a $2.5\,^{\circ}/\mathrm{s}$ swing across the tested temperature range, the z axis only changes within $0.5\,^{\circ}/\mathrm{s}$. It is hoped that this inherent temperature stability will make up for the nonlinear temperature coefficient.

Because of the nonlinear nature of the gyroscope's x-axis temperature drift, the correlation between the linear fitted curve and the actual temperature performance is poor. With a correlation coefficient of

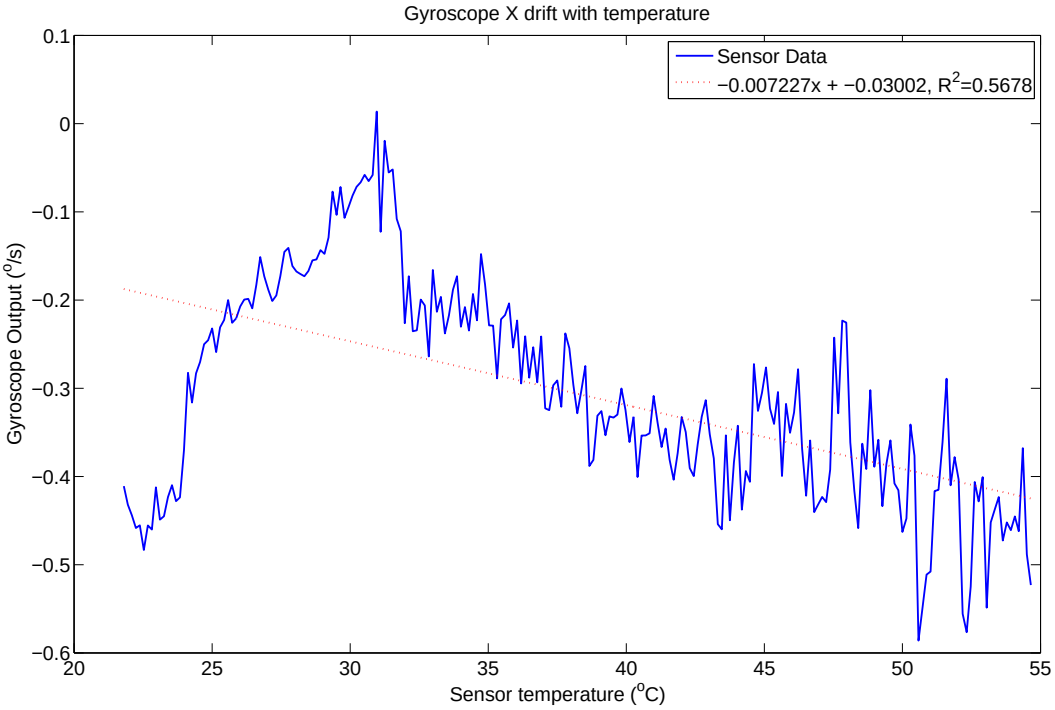

Figure 4.17: Temperature drift of the x-axis gyroscope

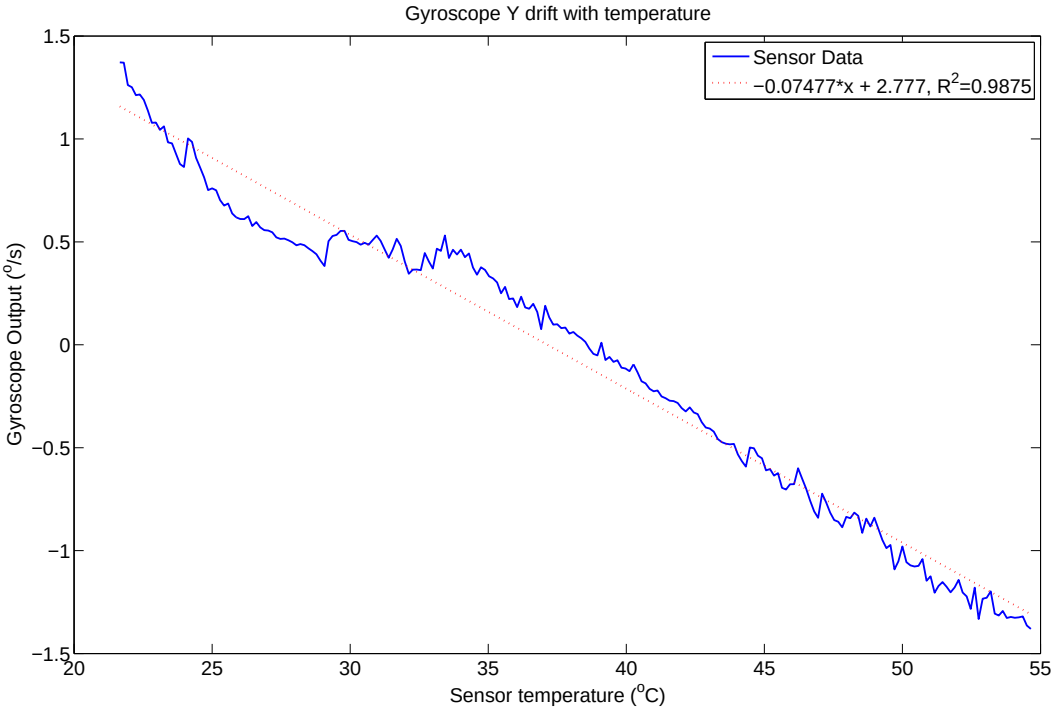

Figure 4.18: Temperature drift of the y-axis gyroscope

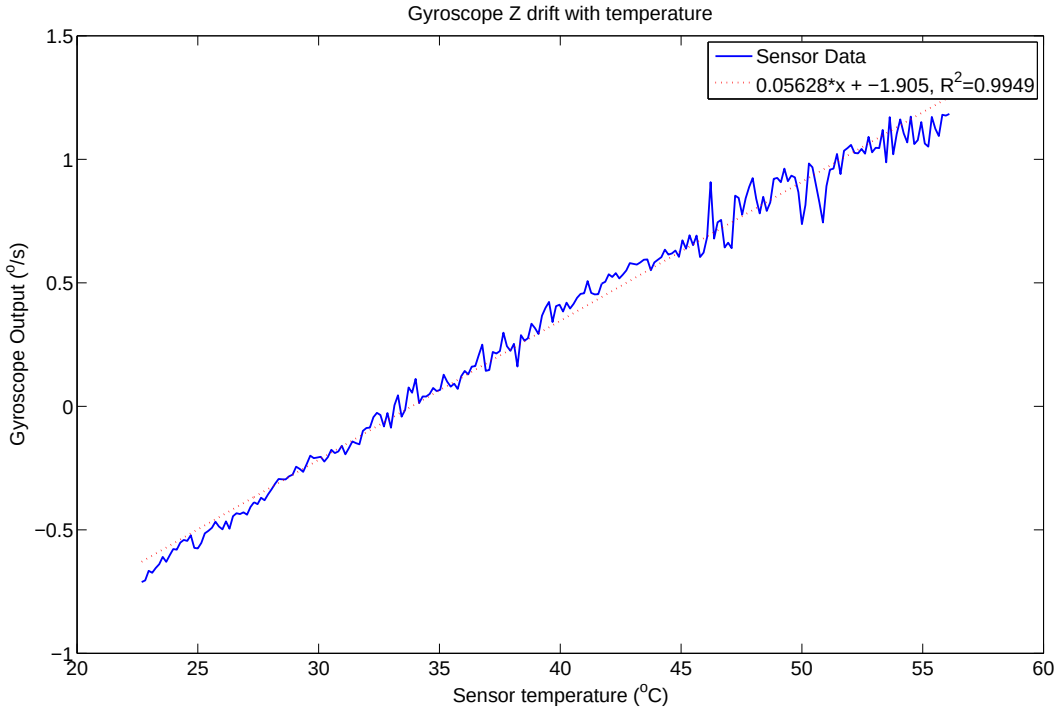

Figure 4.19: Temperature drift of the z-axis gyroscope

only 0.568, it is a value that cannot be trusted. More testing would be required to find out what the true temperature response for this axis is and to find out why it does not respond in a similar fashion to the other axes. One possible reason is that the IMU may have two dual-axis gyroscopes. If that is the case, it is likely that the x-axis is measured internally by two devices and the output provided is the average of these two devices. Sensor fusion of two devices would explain the better performance of the sensor and the fusion algorithm used may explain why the temperature drift is nonlinear.

The y- and z- axes both display very linear temperature response which is shown by the fact that the linear curve fit to the data has correlation coefficients of 0.9875 and 0.9949, respectively. The temperature coefficient given by the ADIS16350 datasheet [29] for all axes is $0.1\,^{o}/s/^{o}C$. The temperature coefficients found experimentally are given in Table 4.7. Each of these coefficients are better than the coefficient provided by the datasheet.

Figures 4.20, 4.21, and 4.22 show the original uncompensated data along with the temperature compensated data for each axis. In each case, the IMU was held stationary and the temperature was changed positively and negatively. As was the case with the accelerometers, the faster the temperature changed, the more nonlinear the temperature drift became. Since the x-axis isn't affected by temperature that much, the temperature compensation doesn't change the data significantly, as is suggested by Figure 4.20. The y- and z-axes, however, benefit greatly from temperature compensation and this compensation has been used for the rest of the measurements in the thesis.

Table 4.7: Actual temperature coefficient of the gyroscopes.

Axis	Temperature Coefficient $(^o/s/^oC)$
x	-0.007
y	-0.075
z	0.056

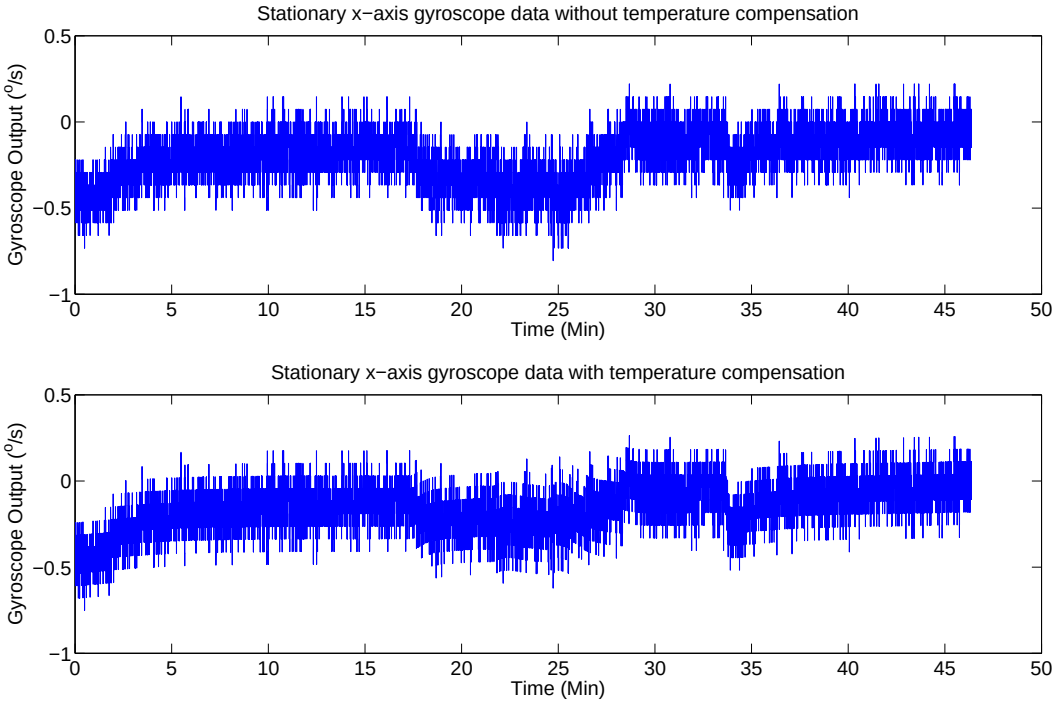

Figure 4.20: Stationary gyroscope data along the x-axis

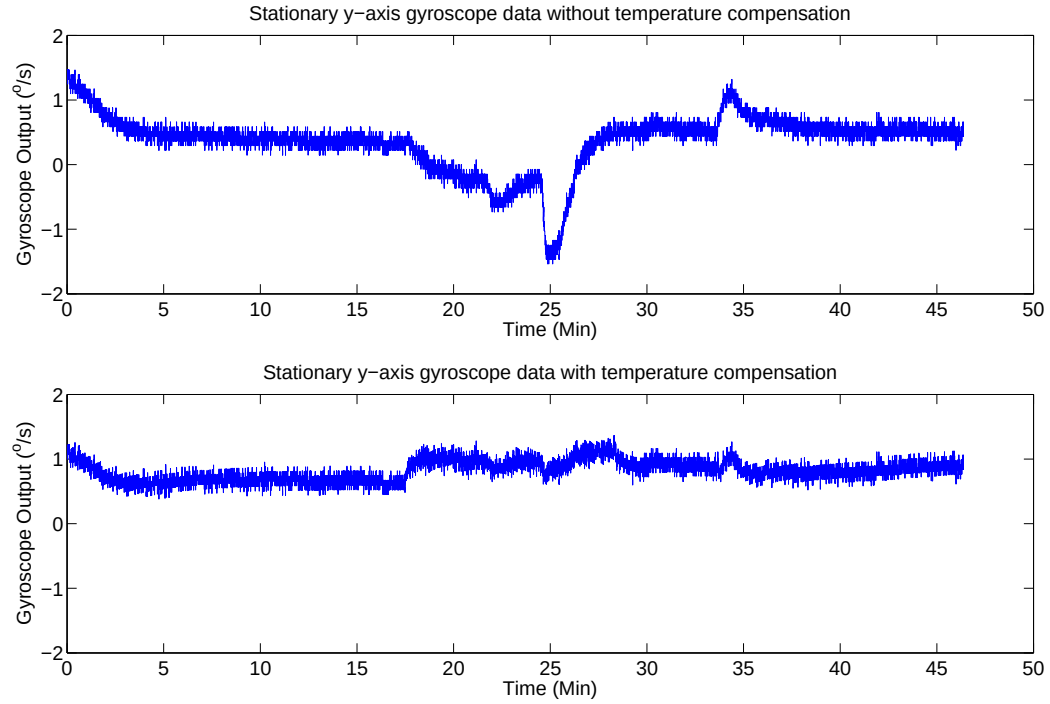

Figure 4.21: Stationary gyroscope data along the y-axis

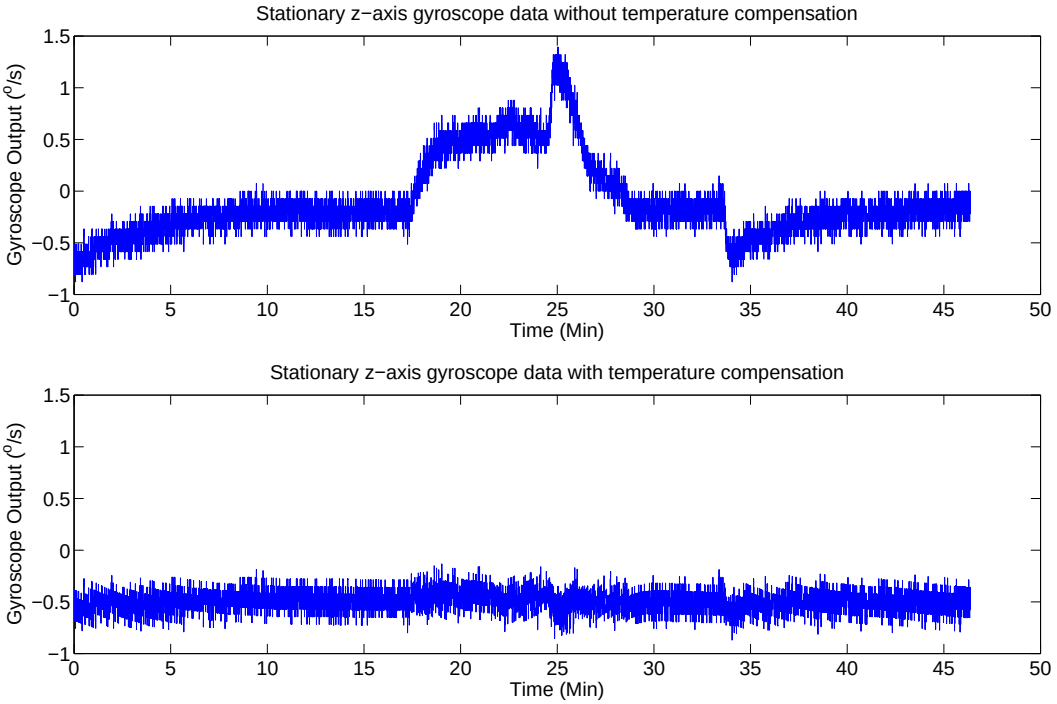

Figure 4.22: Stationary gyroscope data along the z-axis

Gyroscope Offset

Measuring gyroscope offset is much easier than measuring the offset of the accelerometers. The accelerometers are affected by a gravitational force even when stationary but the gyroscopes don't undergo a significant rotation while stationary. Thus, provided that the device has been properly temperature compensated, the average value that each gyroscope puts out while stationary will be the offset associated with that axis. Using the mean of the corrected temperature coefficient data, the offset of each axis was found. The results of these calculations are given in Table 4.8.

During testing, it was found that the offset changed with each cycling of power. This is unfortunate, as it means that the offsets found in Table 4.8 can't be applied in every situation. Those offsets only apply for the set of data that they were calculated from. Each time the device is powered up, the offset needs to be calculated. This requires that the device is held stationary after power up for a short while before samples are taken. For the tracking experiments in this thesis, the device is held stationary for the first 1000 samples and the offset is calculated from this data.

Gyroscope Angular Random Walk

Using the same set of capture data as was used for the accelerometers, the angular random walk of each gyroscope was found using the Allan Deviance method described by Woodman [32]. The graphs in Figures 4.23, 4.24, and 4.25 show the results of the Allan Deviation tests for the gyroscopes. Table 4.9 shows the Angular Random Walk (Allan Deviance) of each gyroscope axis of the IMU.

4.3 Orientation and Position Tracking

In order for any of the error signals to be important or relevant, there must be an end use for the device. The generic algorithm used for strapdown inertial navigation is given in Figure 4.26. This algorithm is used as a base method for calculating the orientation and position of an object, using gyroscopes and accelerometers for the inputs. A significant amount of research has been done on variations of this algorithm by numerous researchers [33–38]. These researchers have chosen and adjusted their filtering algorithms to match specific usage scenarios, with varying degrees of success. We have not come across any research where the author

Table 4.8: Bias of each gyroscope axis.

	Gyroscope Offset
Axis	$(^o/s)$
x	-0.1568
y	0.8228
z	-0.4963

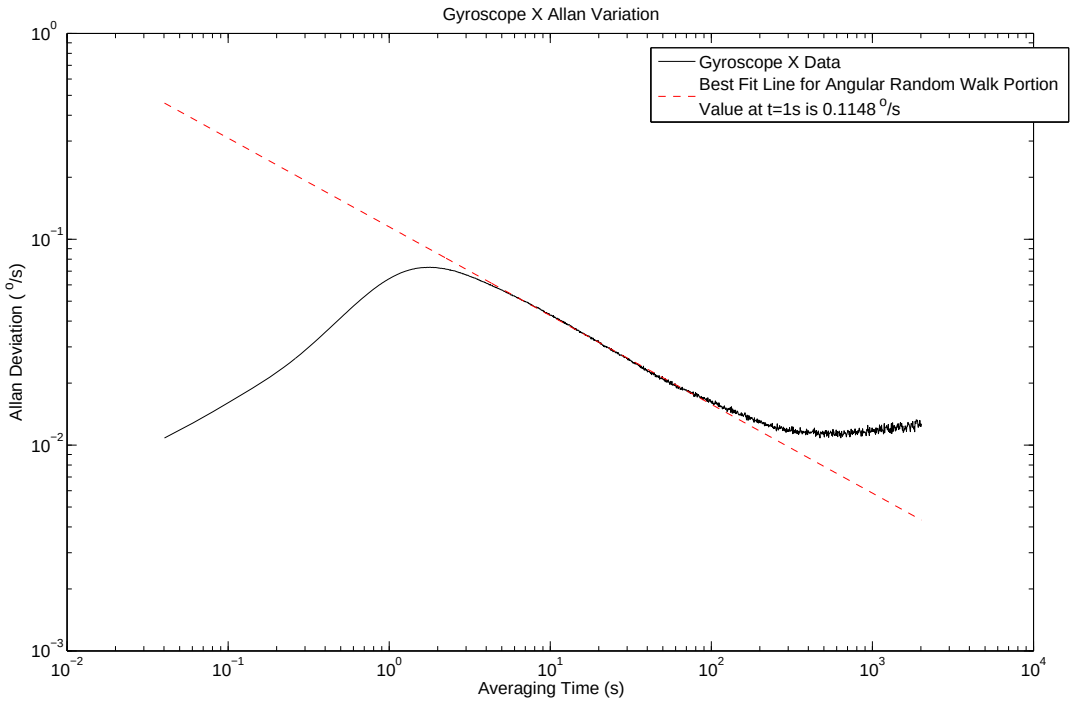

Figure 4.23: Allan Deviation plot of the x-axis gyroscope

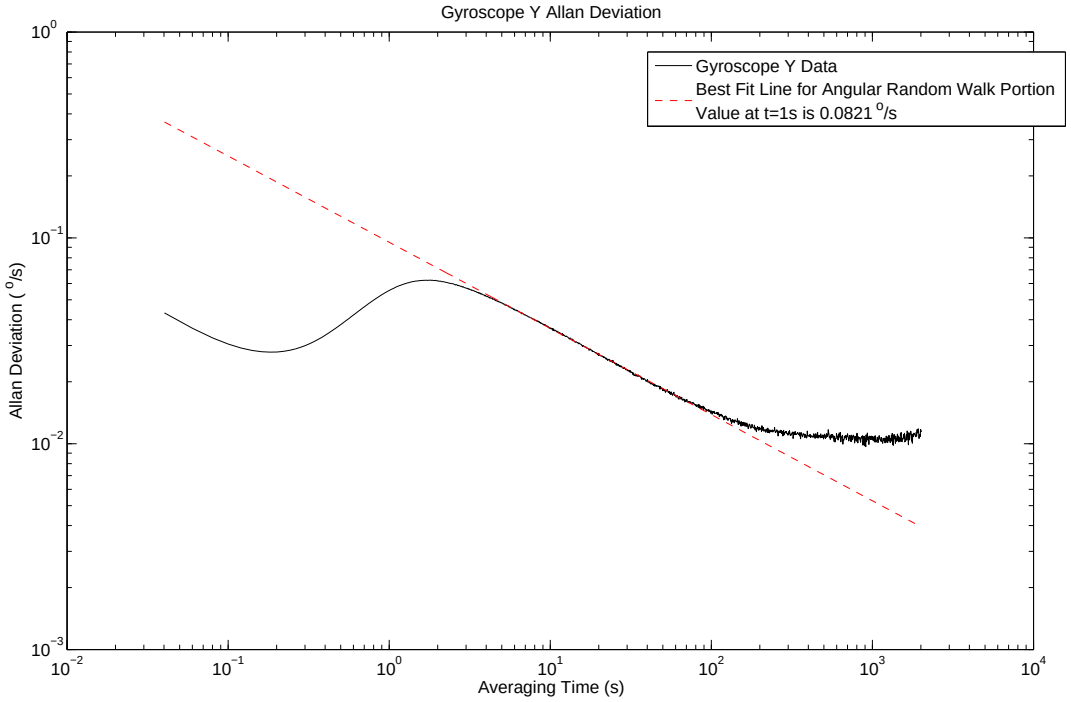

Figure 4.24: Allan Deviation plot of the y-axis gyroscope

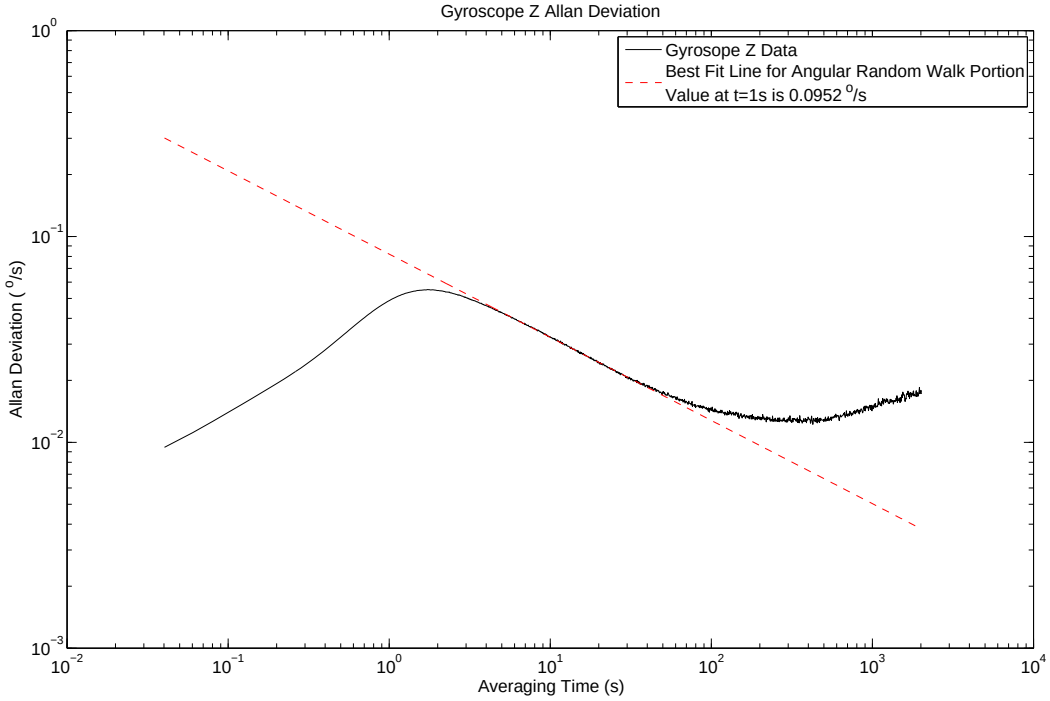

Figure 4.25: Allan Deviation plot of the z-axis gyroscope

Table 4.9: Angular random walk for each gyroscope.

Gyroscope Axis	Angular Random Walk Coefficient $(^o/\sqrt{s})$	Angular Random Walk Coefficient $(^o/\sqrt{h})$
x	0.1148	6.889
y	0.0821	4.928
z	0.0952	5.713

produced a working algorithm which uses only accelerometers and gyroscopes as the input devices, while achieving the type of accuracy necessary to track the motion of an arm for use in a telerehabilitation environment. Most of this research was done between the mid-1990's and the early 2000's, before accurate MEMS devices were readily available. Since then, the availability of inexpensive, accurate, MEMS-based IMU devices has improved greatly. The algorithm in Figure 4.26 was given in [32] and is the basis of the orientation and position tracking algorithm used here.

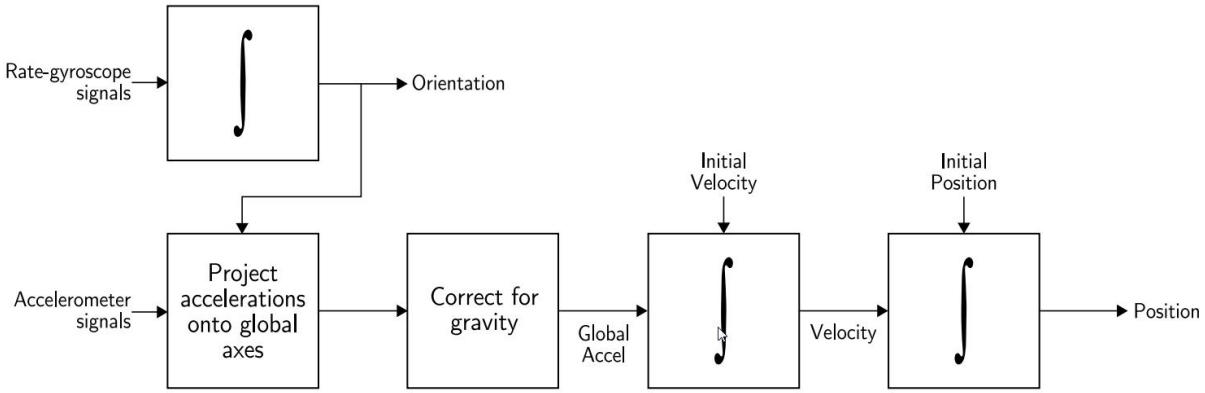

Figure 4.26: Strapdown Inertial Navigation Algorithm

A number of rotation movements were performed with the prototype device and the orientation tracking algorithm was tested. In each case, the device started in a stationary position. The following rotations were performed:

- Rotated 90 degrees around the x-axis and back.
- Rotated 90 degrees around the y-axis and back.
- Rotated 90 degrees around the z-axis and back.
- Rotated 90 degrees around the axis which bisects the x- and y-axes and back.
- Rotated 90 degrees around the x-axis and then 90 degrees around the y-axis and then through the motion backwards.
- Rotated 180 degrees around the y-axis and back.

Showing all of the results produced by these rotations would be onerous and unwieldy, so a single case is focused on here. Since it is easy to describe and demonstrate what forces the sensor undergoes upon rotation and translation, the case where the sensor was rotated 90 degrees around the y-axis is discussed here.

Figure 4.27 shows the raw acceleration which the accelerometers measured upon rotation. Since the acceleration due to gravity is much larger in magnitude than the acceleration that the sensor experienced while rotating , the accelerometer data is essentially a gravity reading. The initial orientation of the sensor

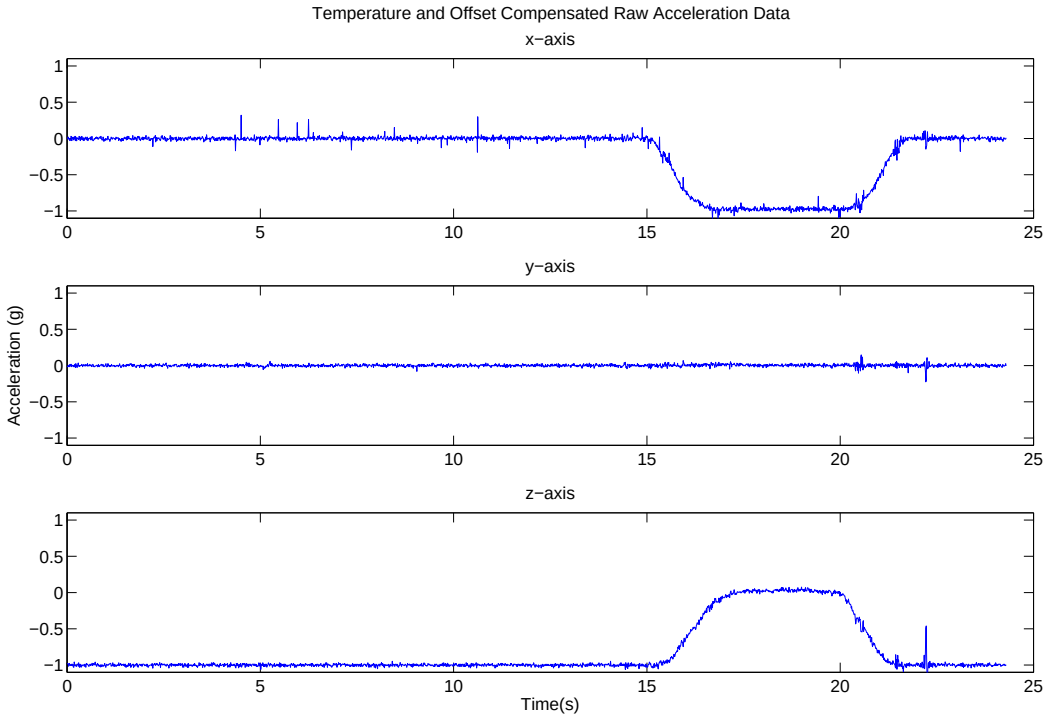

Figure 4.27: Raw Temperature and Offset Compensated Accelerometer Data - Rotate Around y-axis

was with the x- and y-axes perpendicular to gravity and with the z-axis pointing in the opposite direction of gravity (up). As is expected, in this starting orientation, the x- and y-axes showed zero acceleration and the z-axis accelerometer measured -1 g. After the device rotated 90 degrees around the y-axis, the x-axis rotated to face up (away from gravity) and the z-axis rotated to be perpendicular to gravity. The y-axis rotated around itself and as such, came to rest in the same position. After the 90 degree rotation, the z- and y-axis read zero acceleration and the x-axis read -1 g. At $20s$, the sensor was rotated back to the original position. It should also be noted that while the sensor is stationary, the noise in the measurement is consistently in the $\pm 0.05g$ range, with spikes as high as $0.3g$.

The graphs in Figure 4.27 show that the majority of the force experienced by the accelerometers is gravity. In order to see what accelerations correspond to actual movement of the device, the force of gravity must be removed from the measurements. This means that the gravity vector must be tracked in another manner.

The raw angular velocity measurements taken by the three gyroscopes during this sensor motion are shown in Figure 4.28. As should be the case, the x- and z-axes show very little rotation (only shaking of the hand) and the y-axis shows rotation first in the positive direction (rotate z-to-x) then in the negative direction (rotate x-to-z). This angular velocity data can be used to track the rotation of a point from one coordinate space to another. If that point is used to create a vector from the origin, then the angular velocity data can be used to track the movement of a vector as it moves through space.

Multiplying each angular velocity measurement by the sample time 10.38 ms provides the angle which the

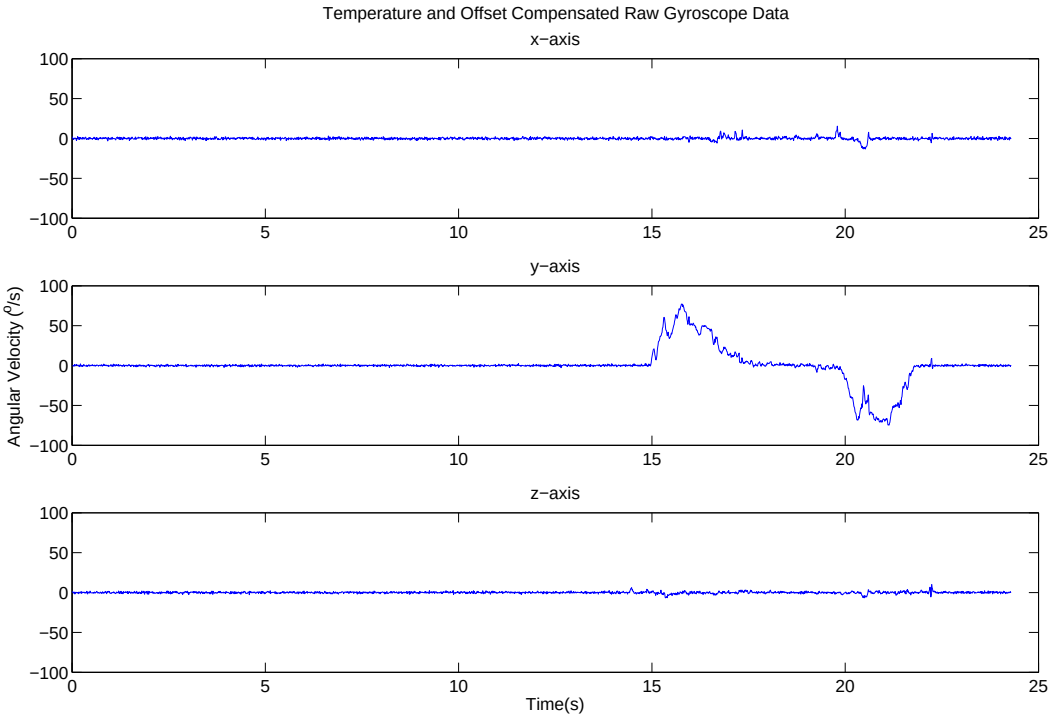

Figure 4.28: Raw Temperature and Offset Compensated Gyroscope Data - Rotate Around y-axis

device has rotated through at each timestep. These angles are necessary to find the quaternion representation of the angular data. The quaternion representation and rotation discussed in Section 3.4.3 allows us to track a vector by following its rotation at each timestep. The obvious vector to track is the gravity vector, so that is what was tracked in this experiment. First, the angles at that timestep are transformed into a quaternion representation of the angles. Then, the gravity vector (obtained from the accelerometers at the beginning of data capture) is rotated from its position at the previous timestep into its new position by quaternion multiplication using the equation $g_t = qg_{t-1}q^{-1}$, where q is the quaternion representation of the angles that the device has rotated through (q^{-1} is its conjugate), g_t is the gravity vector at the current timestep and g_{t-1} is the gravity vector at the previous timestep.

Tracking the gravity vector like this allows us to observe how gravity is being measured by the accelerometers and provides a way to directly remove its effects from the measurement. Figure 4.29 shows gravity as tracked by the gyroscopes. The fact that this graph so closely resembles the raw acceleration data exemplifies the effect that gravity has on the acceleration signal. From this graph, it's obvious that gravity is the largest part of the acceleration measurement. It also shows that the gyroscopes are a reliable way to track orientation with respect to gravity and that the quaternion rotation algorithm is correctly implemented.

Subtracting the gravity data that the gyroscopes provided from the original acceleration data leaves the actual acceleration that the accelerometer experienced. Figure 4.30 shows the result when the gravity data is removed from the acceleration measurement. The graph appears to show no acceleration, even though the

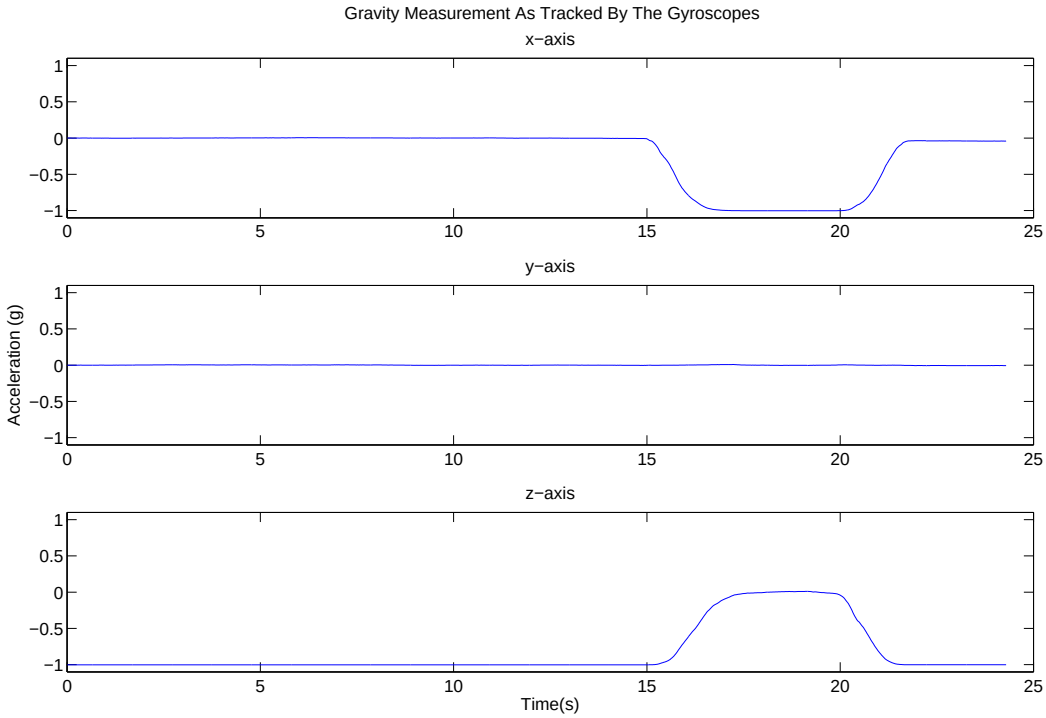

Figure 4.29: Gravity as tracked by the Gyroscopes

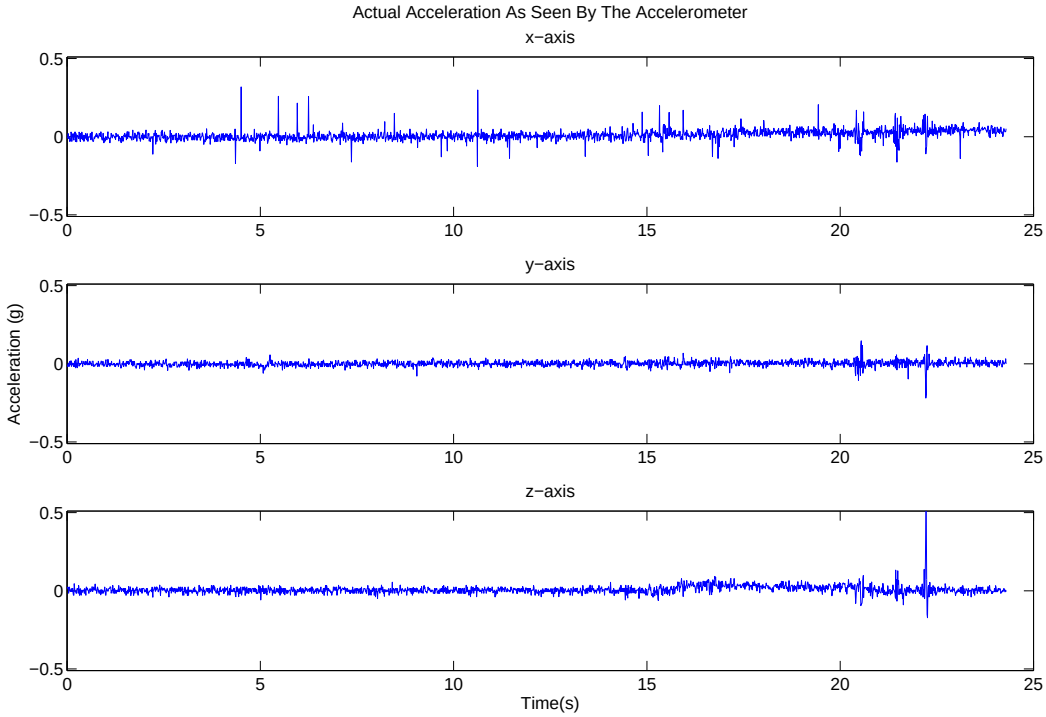

Figure 4.30: Acceleration as experienced by the accelerometers

device was clearly moved. When the sensor was rotated, it moved at least three centimeters in the x- and z-directions. For the sensor to be useful in a prosthetic arm rehabilitation setting, we estimate that the device would need a resolution of at least 1cm and be accurate for a minimum of sixty seconds. This will provide sufficient time to interact in the VE for one training exercise, and will allow reasonably fine motions to be performed (such as picking up a coffee cup in the VE). The graph in Figure 4.30 shows that the majority of the accelerometer signal left after the gravity is removed is noise. Whatever movement happens at slow speeds is hidden by the noise floor, at least visually on the graph. To find out if there is any movement actually measured by the accelerometer, the acceleration data needs to be integrated. If the noise is centred around zero and all of the offset is removed from the accelerometers, the movement should be visible on a plot of the position found by double integrating the accelerometer data. Even if there is a constant offset, the movement should be visible on a plot of the velocity obtained by integrating the acceleration data once. Figure 4.31 shows the velocity obtained from integrating the data in Figure 4.30 and Figure 4.32 shows the position obtained by integrating the velocity data.

As can be seen, when movement starts (at around $15s$), the velocity and position graphs both show a change in state. The velocity of the x- and z- axes begin to diverge from their reasonably constant $0\,\mathrm{m/s}$, which is what is expected. However, movement only occurs for the time periods at $15s - 17s$ and $20s - 22s$. Outside of these times, the velocity should return to zero, whereas the graph obviously does not reflect this. The inaccurate velocity plot could stem from a number of things. First, the offset of the accelerometers may shift upon movement, which might indicate that the moving masses of the MEMS devices do not come to rest in the same position after movement that they started in prior to movement. Second, the gyroscopes may not have a sufficient range to measure quick rotations. If any rotation was faster than $300\,^{\circ}/\mathrm{s}$, it wouldn't be integrated into the gravity tracking calculation and as such would result in the device indicating that it rotated less than it had. This would, in turn, cause the acceleration as seen by the accelerometer measurement in Figure 4.30 to be inaccurate and still contain some portion of the force of gravity in any given measurement. Since the force of gravity is so much stronger than any force exerted on the sensor during movement, any portion of the gravity vector present in the measurement would cause large errors in the acceleration as seen by the device. Closer inspection of Figure 4.30 indicates that there will be an erroneous velocity graph, without actually plotting the graph. It can be seen that the average acceleration of the x- and z-axes during the rotation movements are nonzero. Both axes have a positive average acceleration during movement. Any movement that begins from rest and ends at rest will have an average acceleration of zero over the entire movement. A positive average acceleration would indicate that the velocity increases over the course of the movement, which is exactly what the velocity plot in Figure 4.31 shows. However, the velocity plot does indicate when movement begins, showing approximately zero velocity up until the time when the sensor starts moving, demonstrating that the device's offset changes minimally during rest. This piece of knowledge would be useful for implementing a recalibration algorithm which could be used when the device comes to a rest for some period of time. The fact that the device shows a relatively noisy signal

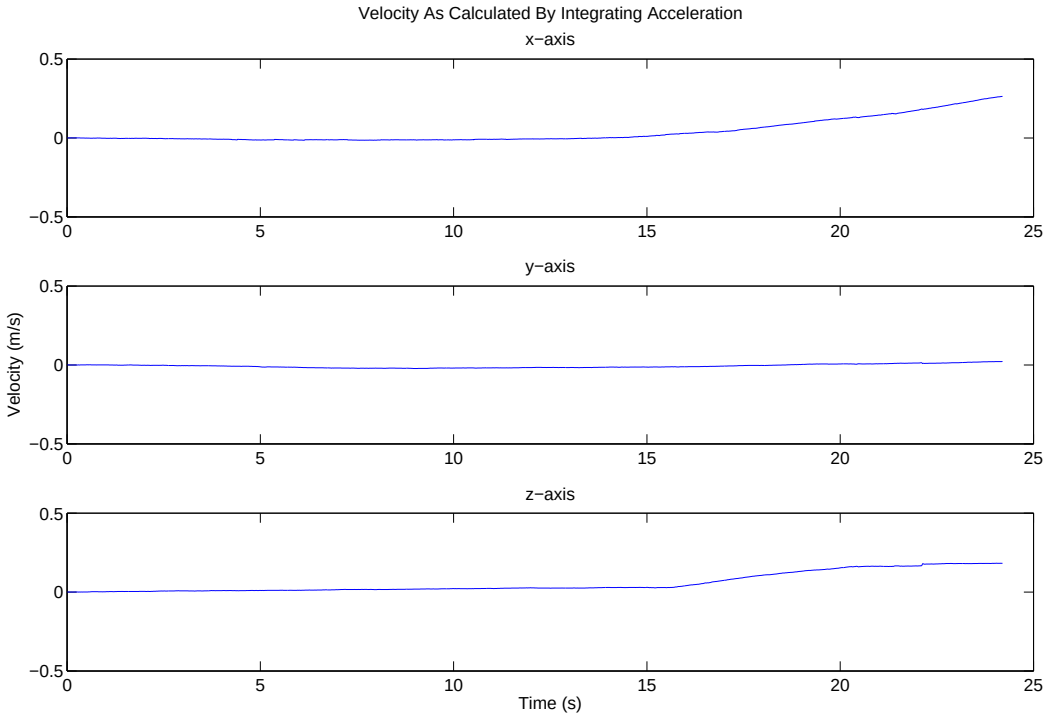

Figure 4.31: Velocity obtained by integrating acceleration

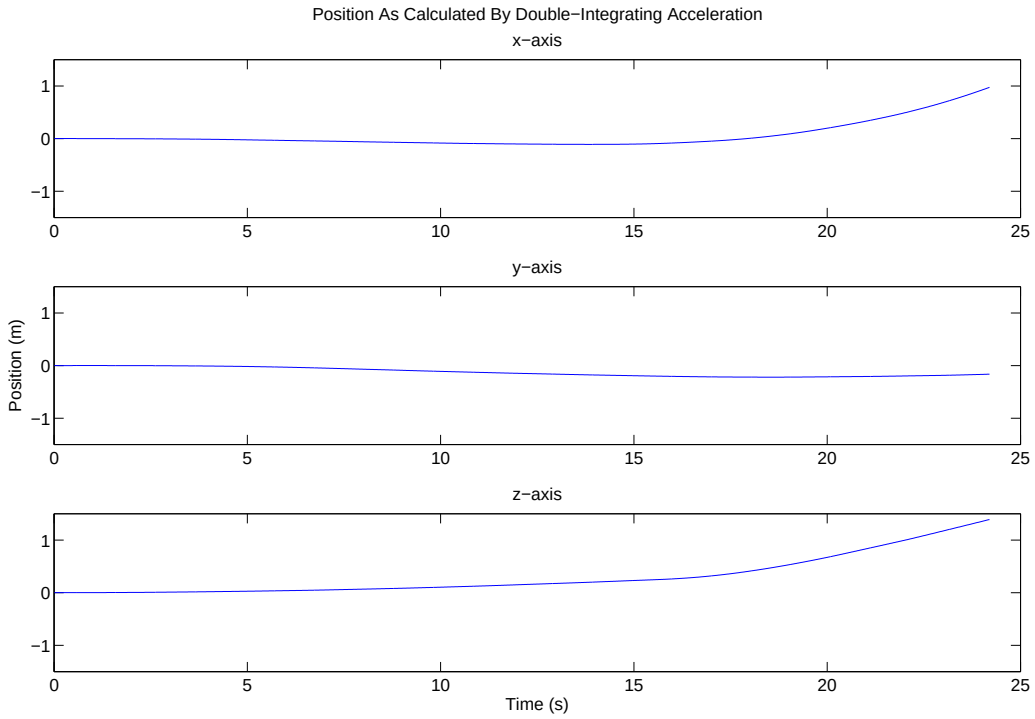

Figure 4.32: Position obtained by integrating velocity

during the initial resting period, yet still integrates to approximately zero for the first fifteen seconds tells us that the noise is centered around zero for that period of time. If an indication of when the movement starts was desired, the slope of the velocity plot would be a much easier way to detect movement than trying to use the acceleration graph.

It is no surprise that the position plot shows the sensor in an incorrect position, due to the errors present in the velocity plot. Any velocity errors are compounded when the data is integrated to give position. The position plot in Figure 4.32 indicates that the device has moved more than 1.4 meters in the positive x-z direction, even though the device actually moved no more than a couple inches at any given time and came to rest in the starting position at the end. This is completely unacceptable for use as a position tracking device for virtual reality rehabilitation of upper extremity amputees. The person would have barely moved their hand and the computer would have told them that they had moved more than double an arm's length away.

Chapter 5

Discussion

The goal for this thesis was to design and implement a small wearable device which would be used for motion capture in a virtual reality rehabilitation environment for upper-extremity amputees. Having a device such as this would provide medical professionals with the ability to more accurately track a patient's progress and would allow them to provide rehabilitation services to patients in sub-optimal situations such as in remote locations or in a war zone situation. This usage scenario prompts a few requirements from the device which may not apply to other motion capture devices. First, the device needs to be small enough to be worn on the arm just above the amputation and lightweight enough to be comfortable without irritating a fresh wound. Since the device is meant to be used as a telerehabilitation device in areas where access to health care is limited, the patient must be able to set up and operate the device without a healthcare professional present. This leads to the requirement that the device be usable in any environment (such as in a patient's home) and that it not have any complicated setup or calibration procedures. Also, the device must capture all of the motions that a patient will make once they are wearing a real prosthetic device. This includes not only translational and rotational movement of the arm, but also the motions which are made by the shoulders to operate the prosthetic's terminal device, such as a hook. The patient must feel like the motion that they perceived themselves making corresponds to what they see in the virtual reality environment. This means that the movements made must be precisely captured and translated to the virtual environment with minimal latency.

5.1 Size and Weight

The prototype device that was designed has a mass of 78g without any batteries. The batteries used in this situation were four AAA batteries, which weigh 100g themselves. This resulted in a total device weight of 178g, which is likely lightweight enough to wear for longer than an hour. The batteries, however, could be worn on a belt clip, which would mean that the device would only weigh 78g near the wound. This is very light and is barely even noticeable when placed on the arm. The final device would be even lighter than this, since it would have a reduction in circuit board size.

The prototype device created measures $65mm$ x $110mm$ x $45mm$ ($2\frac{1}{2}$" x $4\frac{3}{8}$" x $1\frac{3}{4}$"), which is definitely too big to be worn on the arm comfortably. However, the device was designed with large components spaced

out to facilitate prototyping and nearly all of the components used could be replaced by smaller ones, with a more compactly designed PCB layout. The prototype could definitely be redesigned to fit in a footprint of 2" x 2" x 1.5", if not smaller. A device of that size would be more than small enough to wear on an arm comfortably, since there are many personal audio players on the market which are that size and meant to be worn on the arm. Most of them also weigh more than 78g as well. This device can definitely be made small enough to be wearable and comfortable.

5.2 Calibration and Operation

In order to make sure that a patient can use the device without a medical professional present, the calibration procedure must be as simple as possible and the device must be easy to operate.

Many of the motion capture devices in use today have stringent usage scenarios and special setup and calibration procedures. There are many technologies which can increase the accuracy and precision of a motion capture device, but most of them require the device be used in a special environment. GPS integration is one of the most common ways of correcting motion capture information provided by accelerometers and gyroscopes. The problem with GPS is that it requires a line-of-sight connection to a satellite, something that is definitely not available in a person's house or in a large building such as a hospital, so GPS could not be used in this situation.

Another very common way that motion capture is accomplished is to use a camera and marker system. Many movie studios and special effects companies use these systems to capture motion of many objects all at once with great precision. The problem with this system is that it requires a specialized studio with calibrated cameras placed around the room. Not only is the calibration of these systems very complicated, but they also require many pieces of hardware to be set up in precise locations. This definitely doesn't meet our requirement of simplicity, so the camera method was also unsuitable for this thesis.

An inertial measurement system, while less precise than the other two methods discussed here, requires only one device and can be implemented so that the only requirement for calibration is for the device to be kept still for a small amount of time while the system calibrates on its own. Since there is only one device required, the system is easy to ship and easy to wear. All the patient would be required to do would be to strap on the device and follow the instructions in the virtual environment software. This definitely meets our requirement for simplicity and ease of use and is the reason that an inertial measurement system was chosen.

5.3 Response Time

In order to be in any way usable, a motion capture system must be responsive, with a minimum of latency. When the prosthetic arm is displayed in the virtual world, any movements made as perceived by the patient

must match the movements as they are displayed in the virtual world. The kinematics of the prosthetic in the virtual world must also match those of a real prosthetic. Any movements made must also be updated in the virtual world in real-time. If the response time of the virtual world is too slow, then the user will quickly become disoriented and frustrated and less likely to use the device for any length of time.

Each person has their own threshold for what they consider a comfortable latency time between their input and the time they get a response from a feedback device. In this case, this will be the time between the movement of his or her arm and when they see that motion displayed on the screen. An analysis of the lag time requirements of this entire system is outside of the scope of this thesis. However, we can use video game lag and personal experience as a very general guideline. In general, framerates in video games are considered playable if the framerate is over 30fps (33.3ms). Online games tend to be considered playable if the lag time over the internet is 150ms. This means that the acceptable lag time of a playable game is approximately 183ms.

For this device, the sample time of the sensor is 10ms. The wireless UART link between the computer and the sensor is fast enough to keep up with these sample times, so the maximum lag time between the sensor and the data being stored in the computer is 20ms. After the data is stored in the computer, it needs to be filtered and the data is then used to calculate position and orientation. The algorithm used will define what the total latency is in the system. While none of the algorithms in this test were used in real-time with the sensor, calculations on each dataset took a fraction of what the total sample time of that dataset was. The data provided by a static device over the course of a few hours was used to calculate the position. This calculation took only a few minutes, indicating that all of the algorithms provided here can be calculated in real-time with a sample period of 10ms. Thus, the algorithm only adds a couple of milliseconds on top of our 20 millisecond worst-case scenario. Any filter implemented will be responsible for the most latency. If a simple averaging filter is used, then there will be an extra 10ms of latency added for each increase in sample number of the filter. An averaging filter length of five samples will have a minimum latency of 60ms. However, a recursive filter such as a Kalman filter, will only have a latency of 30ms, since it requires only one state of past information to calculate the current data points.

With calculation overhead and the required sample time for filtering, the total time between when a movement is made and when the position information is passed to the virtual world can be kept to approximately 35ms, depending on the filter chosen. This leaves another 150ms for the virtual world to use the data and render the world on the screen. It is expected that this is more than enough time, since even the most complicated and detailed of video game worlds can be rendered at real-time framerates of 60fps (16.6ms) with the current generation of video card technology. It is expected that this device can easily be implemented so that the time between a users input and the screen displaying that motion is kept to under 100ms, which is within our estimated acceptable latency.

5.4 Accuracy and Precision

The biggest hurdle to designing a compact motion capture system using only inertial measurement devices is achieving an acceptable level of accuracy and precision. The device must be precise enough to measure small accelerations repeatedly without any significant drift and the measurements must be accurate enough to provide a correct measurement of that acceleration value. Since so many integrations are required within the system, any precision or accuracy errors will cause significant issues in the final result, with the error getting worse the longer that the system is in use.

5.4.1 Linear Displacement Sensor

The linear displacement sensor is a direct measurement device. Each measurement is directly used and observed. Therefore, the required accuracy of this sensor is the same as the required accuracy of the result. Experimentally, the linear displacement sensor was found to be accurate to within less than $\pm 5\%$ of the measured value in all cases and had a resolution of 0.03mm. There was an issue with sensor precision, as the zero point changed between each movement. It was measured that the zero point changed across a span of 1.7mm. This is attributed to the fact that the prototype was very crudely held together with tape and had a significant amount of play in it. If a final device was made with rigid components, there would be much less error in this measurement. The change in zero point would be perceptible to the user, as it would show up in the virtual world as the hook being slightly open (up to 1mm). This wouldn't be an issue in practice because in this situation, the zero point could be held fixed programmatically and the user wouldn't perceive the error in measurement each time the sensor came back to rest. This error wouldn't be additive and instead is just a random amount of error on the rest point of the terminal device.

It was decided that as long as the terminal device measurements were accurate within 2mm, the user would be able to use the system without any trouble. While the zero point (closed terminal device) changed with each measurement, the hook opening that the device measured was always within 5% of the actual opening and thus was a very accurate measurement of terminal device position. If the virtual world was designed to check for a return-to-close situation, then the user would likely have no idea that there was any error in the measurement at all. For this reason, it was decided that the linear displacement sensor was suitable and sufficient for measuring the movements made by the shoulders for the purpose of simulating a terminal device.

5.4.2 Orientation Measurements

When interacting with the virtual world, movements made in the real world must translate well to movements in the virtual world. The system will be unusable if the patient moves their arm to the left, but the virtual world screen shows the arm moving up. The orientation measurement provided by the gyroscope must be accurate enough so that the person using the system doesn't become disoriented. A user will not likely notice

if the arm in the virtual environment is rotated one or two degrees differently than their arm actually is, but five degrees would likely be noticed. In the past, MEMS gyroscopes were not readily available. In the last few years, cheap and accurate MEMS devices have become very common. The gyroscopes were tested so that an error profile for these devices could be found. The error properties of the gyroscopes provide some insight as to whether or not the devices chosen are suitable for measuring orientation. The relationship between temperature and the output of the gyroscopes was found first. It was found that temperature had a significant effect on the gyroscope outputs. The z-axis was the worst, having a temperature coefficient of $0.056^o/s/^oC$. If the gyroscope output was used to find orientation over a period of one minute, a difference of ten degrees celsius would mean that the error in orientation at the end of that sixty seconds would be 33^o. Obviously, this would be an unacceptable result so the gyroscope output was temperature compensated.

A stationary offset in the gyroscopes' outputs would also cause an error in orientation which would grow linearly with time. The gyroscope offset was found to change with each power cycle of the device so it is required that the stationary offset be tested every time the device was turned on. This means that the device must be held stationary for a calibration period each time it is used. When tested, the maximum bias amongst the three gyroscopes was found to be $0.8228^0/s$. Over the course of one minute, this would correspond to an error in orientation of 49.3^o, which again, would obviously be unacceptable if uncorrected.

The stationary bias and the temperature offset are easily corrected in the orientation algorithm, but the angular random walk is much more difficult to remove, due to its random nature. The gyroscopes were tested and it was found that the worst case angular random walk was found to be $0.1148^o/\sqrt{s}$. The angular random walk describes how the standard deviation of the output changes with time. After sixty seconds, the standard deviation of that gyroscope output would be 0.889^o. What this means is that after one minute, the orientation provided by the gyros will have an error of 0.889^o, caused by white noise within the gyroscope. As was discussed earlier, an error of one degree after one minute will not likely be noticeable by the user and is acceptable for use in this motion tracking system. Provided that the temperature dependence and the stationary bias are always addressed in the orientation algorithm, the gyroscopes used in this motion tracking system will provide an acceptable orientation measurement. During the course of this thesis project, this was found to be true when the gyroscope measurements were used to accurately correct the orientation of the measurements provided by the accelerometers.

5.4.3 Position Measurements

A useful virtual rehabilitation environment would display changes in position that are perceived by the patient as being the same change in position that they made in real life. Since rehabilitation would include exercises such as teaching the patient to use a prosthetic device to pick up an object such as a coffee cup, the virtual world would need to be able to display the position with a minimum of 1cm accuracy. Better accuracy in this case would certainly be helpful, but the exercises would be beneficial even if the position that the arm is shown in is 1cm different from what it is in real life. The incremental change in position in

this case would be much more important than the absolute position. As long as the person saw that they were a certain distance away from an object on the screen and a movement of that distance in real life was well translated to the virtual environment, the exercise would be successful. This also means that if the patient is not moving, it is important that the device on screen be shown as not moving.

Since the accelerometers are MEMS devices much like the gyroscopes used in this device, the accelerometers have similar error properties as the gyroscopes. What separates the accelerometers in this case is that the error is compounded exponentially in the position output, rather than linearly like they are in the orientation output. The result of this is that the tolerance for error in the accelerometers is much smaller than that of the gyroscopes.

Temperature and stationary bias affect the accelerometers in the same way as they affect the gyroscopes. The worst temperature performance of the accelerometers was given by the z-axis accelerometers. The output of this device changed by 5.98 mg/°C (0.0587 m/s^2). A 10 degree change in the temperature of the device would result in a change in the output of the accelerometer of 0.587 m/s^2. Over 60 seconds, that would result in an added position error of 2.81 km over the same measurement 10 °C cooler!!

The worst stationary bias found for the accelerometers was the y-axis, with an offset of 18.51 LSBs; which translates to 0.458 m/s^2. Over one minute, this would result in a change in position of 1.65 km. It is apparent from the temperature and bias errors that a small amount of error in acceleration results in a large position error. It is an absolute must that the temperature and bias errors are removed from the acceleration measurements if a reasonable result is ever to be expected.

Since they are MEMS devices, the accelerometers are affected by white noise in the same manner as the gyroscopes are. When dealing with accelerometers, angular random walk is called velocity random walk and allows us to find the standard deviation of the velocity after a given length of time. The worst velocity random walk found in the accelerometers was $0.0028 m/s/\sqrt{s}$. After one minute, the velocity provided by the device would have a standard deviation of 0.0217 m/s. An error of 0.0217 m/s would result in an error in position of 1.3m after one minute.

In Section 4.3, the position of a stationary sensor was found by integrating the output of the accelerometer twice. Since the sensor was stationary, the position after a given time provides the error in the position at that time. The accelerometers' outputs were first corrected by the orientation given by the gyroscopes. This adds error into the acceleration data in addition to the errors which are inherent in the measurements. The data showed that the device had moved approximately 1.5 m in less than 25 s. This shows that the error characteristics of the devices found in this project are accurate and correctly estimate the performance of the device. However, it also shows that the algorithm used here is insufficient for the tracking of position of a device.

Chapter 6

Conclusion and Future Work

6.1 Summary and Conclusion

In this thesis, we researched commercially available sensors in order to determine their suitability when combined as a motion tracker for use in a virtual rehabilitation environment.

The device needs to be portable enough to be easily sent to remote locations. This imposes size and weight restrictions. The prototype designed implies that the final device can easily be made as small as 2" x 2" x 1.5" and the prototype weighs less than 80 grams, which is definitely small and lightweight enough to be shipped anywhere in the world at minimal cost. The size and weight of this device also satisfies the requirement that it be comfortable to wear for extended periods of time.

In order for the device to be used by a patient without a doctor or technician present, it needs to be very easy to set up and use without complicated calibration procedures. The algorithm used by the device created here only requires the patient to hold still for a few seconds at power up and periodically throughout the therapy sessions. Calibration of the device is performed in the algorithm without any other user interaction. This satisfies the simplicity requirement.

For the device to be useful to the patient, it must be responsive with minimal latency. For it to be useful to the physician, it must provide data which can be sent in real time over the internet. The sensor provides data at a rate which can be transmitted over a serial port connection which is far slower than the internet connections available around the world today. This implies that the data provided by the sensor could be sent in real-time over an internet connection with no interruptions or data stream issues. The hardware and software for the sensor together provide measurement data within $35ms$ of the corresponding movement. This is fast enough to provide a pleasant user experience to the patient.

Table 6.1 shows pass/fail information for the devices requirements if it is to be used as a motion tracker. Of all the requirements for the sensor, the most critical is that the sensor be precise and accurate for the duration of a training exercise. While the accelerometers fail in their position measurement, the majority of the components of the motion tracking sensor designed here satisfy their individual requirements. The linear displacement sensor provides accurate hook position data. The error of the linear displacement sensor is stable and doesn't increase with time, so it will always provide accurate hook opening data. The gyroscopes provide orientation data which are accurate to within one degree over sixty seconds. This is sufficient for

orientation tracking over short durations, such as that of one rehabilitation exercise. Periodic recalibration, however, will be required when the sensor is used for extended periods of time. The the data provided by the accelerometers are most susceptible to accuracy and precision issues. When used as proposed in this thesis, the accelerometers provide an error in measurement of $1m$ over less than thirty seconds. This will be unacceptable to a user and will make the device unusable for position measurement. However, the device would be very useful for orientation tracking and hook cable emulation.

Table 6.1: Device Requirement Results

Requirement	Value	Pass / Fail
Size	<2" x 2" x 1/5"	PASS
Weight	78g	PASS
Calibration	Automatic	PASS
Response Time	35ms	PASS
Linear Displacement Sensor	$\pm 5\%$	PASS
Orientation	0.0889^o / 60s	PASS
Position	>1m / 30s	FAIL

The research in this thesis has provided some answers as to the suitability of this device as a motion tracker. In the end, the device created here for use as a motion capture device and the algorithm employed is an acceptable orientation tracker and hook emulator, meeting those portions of the requirements of a virtual reality prosthetic rehabilitation environment. However, more work must be done in order to use the device as a position tracker as is intended here. A number of algorithm substitutions and improvements can be implemented and some existing issues can be solved with more time and research. Performing some of the fixes and substitutions outlined in Section 6.2 will give a more complete picture of the device's feasibility.

6.2 Improvements in Future Versions of the Device

If we were to attempt to create this device again, some of the things that we would do differently would be use better sensors, implement better filtering, and (likely most significantly) attempt to find another method of correcting position. What follows here is a discussion on some of these new implementations and what effect they may have on the result.

6.2.1 Improved MEMS Devices

In 2008, when this research was first started, MEMS IMUs were just starting to be released commercially as a unit. At the time, the devices which were available for purchase were designed primarily for automotive applications. The intended use for the IMU device was to capture movement data of a vehicle during braking,

acceleration and accidents. This meant that the acceleration range of the device was large ($\pm 10\,\mathrm{g}$) and the gyroscopic range of the device was small ($\pm 300\,^\circ/\mathrm{s}$), since vehicles tend to go through large accelerations and slow rotations. A human arm, on the other hand, tends to be subjected to smaller accelerations and quicker rotations; especially in a rehabilitation setting. If the ranges of the accelerometers in the IMU were smaller (closer to $\pm 2\,\mathrm{g}$), the resolution of the device would have been better, allowing a more accurate tracking of distance moved. The ranges of the gyroscopes in the IMU provided are too small for the application. A device which has a range of at least $\pm 1200\,^\circ/\mathrm{s}$ for each gyroscope would be more suited to this application.

In order to make the final device as small and unwieldy as possible, the IMU used should be as small as possible. At the beginning of the project, the smallest triaxial IMU device that could be found was approximately 2" × 1" × 1". A smaller IMU would not necessarily improve the accuracy of the device, but could prove to be more suitable in terms of wearability.

In 2008, only a small handful of automotive IMU devices were available. Now, a wider range of sensors can be obtained, which may improve the performance of this device. Newer devices provide better accuracy and resolution with better noise characteristics, in a smaller package. They also come with algorithmic tweaks, such as automatic temperature calibration, to help improve the noise performance of the devices. For example, Analog Devices' ADIS16334 has a smaller package at 1.25" × 1" × 0.5". Their ADIS16480 provides gyroscope output of $\pm 450\,^\circ/\mathrm{s}$, which would be an improvement on the current $\pm 300\,^\circ/\mathrm{s}$ in the project device. That IMU also has accelerometer output of $\pm 5\,\mathrm{g}$, which would help, and has much better noise characteristics (including temperature compensation) than the device used in this project. Another useful feature of the ADIS16480 is that it can provide Euler outputs, making the algorithm used in finding the device's orientation much more straightforward and more accurate than the algorithm used in this thesis. A more modern IMU is likely to improve the end result and may improve it enough to make the device feasible as a motion tracker.

6.2.2 New Functionality

Using only accelerometers and gyroscopes to track positioning has proven to be somewhat limiting in terms of the accuracy of this device. Since accelerometers are used to improve the accuracy of the gyroscope readings, rotations around the gravity vector are less accurate than rotations around other arbitrary axes. Implementing a third redundant measurement of orientation should greatly improve the orientation measurements, especially when rotating around the gravity vector. One of the most commonly implemented redundant measurements in this case is magnetic field measurement. While a magnetic field measurement suffers the same accuracy problem around the axis of magnetic north as the accelerometers have around the gravity vector, implementing triaxial magnetic field measurements and fusing that data with the gyroscope and accelerometer measurements will improve the overall orientation measurement and the accelerometers and magnetometers will offset each others' inadequacies. A more accurate orientation measurement will result in a more accurate position measurement. Newer IMUs from Analog Devices, such as the ADIS16480,

also include magnetometer readings.

While adding magnetometers will certainly improve the orientation and the position measurements, a direct measurement of position would be the most significant improvement that could be made to the device. Hardware implementations of a direct position measurement that still met the project requirements (small, lightweight and no advanced calibration requirements) are possible, but are likely expensive and computationally complex. For example, adding three axes of ultrasonic transceivers could theoretically map out the room that the patient is standing in and consistently provide a measurement of where the device is located in the room. In order to achieve this, the room mapping algorithm would be very complex and may or may not be possible in real time. Optical pattern recognition may also be implemented using very small mounted cameras, provide that such a device is available. Mapping the room through optical pattern recognition algorithms would likely require as much or more processing and algorithmic power as the ultrasonic possibility, but the CCD of a digital camera may prove to be a more reliable and easily interfaced device than ultrasonic transceivers. Further research into these options would need to be done and are far out of the scope of this project. Each option would likely result in a full thesis project in and of itself.

6.3 Alternative Algorithms

In this thesis, we implemented quaternion rotation in order to track the orientation of the sensor. This rotation algorithm works very well but it doesn't implement any filtering. There are a number of filters which may improve the result of the algorithm, but the one which is most likely to result in the best improvement is a Kalman filter. The Kalman filter can be implemented in a number of ways and there are many different versions of that filter. One of the best things about the Kalman filter is that it's recursive and only requires one previous state in order to work. This keeps the latency to a minimum and allows for much faster real-time operation - perfect for implementation in a project such as this one. Any future versions of this project should look very seriously into implementing a Kalman filter of some sort.

References

[1] D. S. Childress, "Historical aspects of powered limb prostheses," *Clinical Prosthetics & Orthotics*, vol. 9, no. 1, pp. 2–13, 1985.

[2] Amputee Coalition of America - NLLIC staff, "Amputation statistics by cause - limb loss in the united states." `http://www.amputee-coalition.org/fact_sheets/amp_stats_cause.html`, 2008. Accessed Nov. 2012.

[3] A. Esquenazi MD and R. H. Meier III MD, "Rehabilitation in limb deficiency. 4. limb amputation," *Achives of Physical Medicine and Rehabilitation*, vol. 77, pp. S18–S28, March 1996.

[4] D. M. Kerkovich, "A report on the amputee healthcare and prosthetics workshop sponsored by walter reed army medical center and the department of veterans affairs (va)," *Journal of Rehabilitation Research & Development*, vol. 41, pp. xiii–xv, May/Jun 2004.

[5] J. Crosbie, S. McDonough, S. Lennon, and M. McNeill, "Development of a virtual reality system for the rehabilitation of the upper limb after stroke," *Studies in Health Technology and Informatics*, vol. 117, pp. 218–222, 2005.

[6] M. Kuttuva M.S., R. Boian Ph.D., A. Merians Ph.D. P.T., G. Burdea Ph.D., M. Bouzit Ph.D., J. Lewis M.S., and D. Fensterheim, "The rutgers arm, a rehabilitation system in virtual reality: A pilot study," *CyberPhsycholgy and Behavior*, vol. 9, no. 2, pp. 148–152, 2006.

[7] J. E. Deutsch and A. Mirelman, "Virtual reality-based approaches to enable walking for people post-stroke," *Topics in Stroke Rehabilitation*, vol. 4, pp. 44–53, Nov-Dec 2007.

[8] A. Gaggioli PhD, A. Meneghini MD, F. Morganti PhD, M. Alcaniz PhD, and G. Riva PhD, "A strategy for computer-assisted mental practice in stroke rehabilitation," *Neurorehabilitation and Neural Repair*, vol. 20, no. 4, pp. 503–507, 2006.

[9] A. S. Merians PhD PT, H. Poizner PhD, R. Boian PhD, G. Burdea PhD, and S. Adamovich PhD, "Sensorimotor training in a virtual reality environment: Does it improve functional recovery poststroke?," *Neurorehabilitation and Neural Repair*, vol. 20, no. 2, pp. 252–267, 2006.

[10] R. Kizony MSc, L. Raz MSc, N. Katz PhD, H. Weingarden MD, and P. L. (Tamar) Weiss PhD, "Video-capture virtual reality system for patients with paraplegic spinal cord injury.," *Journal of rehabilitation research and development*, vol. 42, pp. 595–608, September/October 2005.

[11] S. L. Whitney, P. J. Sparto, L. F. Hodges, S. V. Babu, J. M. Furman, and M. S. Redfern, "Responses to a virtual reality grocery store in persons with and without vestibular dysfunction," *CyberPsychology & Behavior*, vol. 9, no. 2, pp. 152–156, 2006.

[12] Y. Baram PhD and A. Miller MD PhD, "Virtual reality cues for improvement of gait in patients with multiple sclerosis," *Neurology*, vol. 66, pp. 178–181, January 2006.

[13] H. Tanie, K. Yamane, and Y. Nakamura, "High marker density motion capture by retrore?ective mesh suit," in *IEEE International Conference on Robotics and Automation*, (Barcelona, Spain), pp. 2884–2889, 18–22 April 2005.

[14] K. Yamane, T. Kuroda, and Y. Nakamura, "High-precision and high-speed motion capture by combining heterogeneous cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, (Sendai, Japan), pp. 279–286, September 28 – October 2 2004.

[15] J. C. Barca, G. Rumantir, and R. K. Li, "A new illuminated contour-based marker system for optical motion capture," *Innovations in Information Technology*, pp. 1–5, November 2006.

[16] A. Sundaresan and R. Chellappa, "Markerless motion capture using multiple cameras," *Computer Vision for Interactive and Intelligent Environment*, pp. 15–26, November 17–18 2005.

[17] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, pp. 585–601, 2003.

[18] Meta Motion, "`http://www.metamotion.com`." Accessed Dec. 2012.

[19] A. Menache, *Understanding motion capture for computer animation and video games.* Morgan Kaufmann, 1 ed., March 17 2000.

[20] Polhemus, "`http://www.polhemus.com`." Accessed Dec. 2012.

[21] A. B. Chatfield, *Fundamentals of High Accuracy Inertial Navigation*, vol. 174 of *Progress in Astronautics and Aeronautics*, ch. Preface, p. xiii. American Institute of Aeronautics, Inc., 1997.

[22] H. Vajihollahi, H. Shateri, and S. Siahpoushan, "Mems gyroscopes." `http://knol.google.com/k/hamid-vajihollahi/mems/2g44ddbae4k1v/2`. Accessed Sep. 2010.

[23] K. S. Kruszelnicki, "Faucault's pendulum." `http://www.abc.net.au/science/k2/pendulum/pendulum.htm`. Accessed Dec. 2012.

[24] R. M. Rogers, *Applied Mathematics In Integrated Navigation Systems.* AIAA Education Series, American Institute of Aeronautics and Astronautics, Inc., 2nd ed., 2003.

[25] J. Vince, *Mathematics for Computer Science.* Springer London, 2nd ed., 2006.

[26] M. Baker, "Maths - axis and angle." `http://www.euclideanspace.com/maths/geometry/rotations/axisAngle/index.htm`. Accessed Dec. 2012.

[27] W. R. Hamilton, "On quaternions; or on a new system of imaginaries in algebra," in *Philisophical Magazine*, 2000.

[28] Sparkfun Electronics. `http://www.sparkfun.com`, note = Accessed Dec. 2012,.

[29] Analog Devices, Inc., *Analog Devices ADIS16350 Datasheet*, rev. a ed., 2008.

[30] E. A. Coutsias and L. Romero, "The quaternions with an application to rigid body dynamics," tech. rep., Department of Mathematics and Statistics, University of New Mexico, February 1999.

[31] MTS Systems Corporation, *MTS Tempotronics Magnetorestrictive Linear-Position Sensors C-Series Core Sensor Product Specification*, rev. h ed., 2007.

[32] O. J. Woodman, "An introduction to inertial navigation," Tech. Rep. 696, University of Cambridge Computer Laboratory, August 2007.

[33] J. Marins, X. Yun, E. Bachmann, R. McGhee, and M. Zyda, "An extended kalman filter for quaternion-based orientation estimation using marg sensors," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 4, pp. 2003–2011, 2001.

[34] X. Yun, M. Lizarraga, E. Bachmann, and R. McGhee, "An improved quaternion-based kalman filter for real-time tracking of rigid body orientation," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, pp. 1074–1079, oct. 2003.

[35] Q. Wang, C. Rizos, Y. Li, and S. Li, "Application of a sigma-point kalman filter for alignment of mems-imu," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pp. 44–52, May 2008.

[36] K. Kobayashi, K. Cheok, and K. Watanabe, "Estimation of absolute vehicle speed using fuzzy logic rule-based kalman filter," in *American Control Conference, 1995. Proceedings of the*, vol. 5, pp. 3086–3090, Jun 1995.

[37] W. T. Ang, P. Khosla, and C. Riviere, "Kalman filtering for real-time orientation tracking of handheld microsurgical instrument," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2574–2580, Sept.-Oct. 2004.

[38] J. Hall, N. Knoebel, and T. McLain, "Quaternion attitude estimation for miniature air vehicles using a multiplicative extended kalman filter," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pp. 1230–1237, May 2008.

Appendix A

First Prototype Schematic

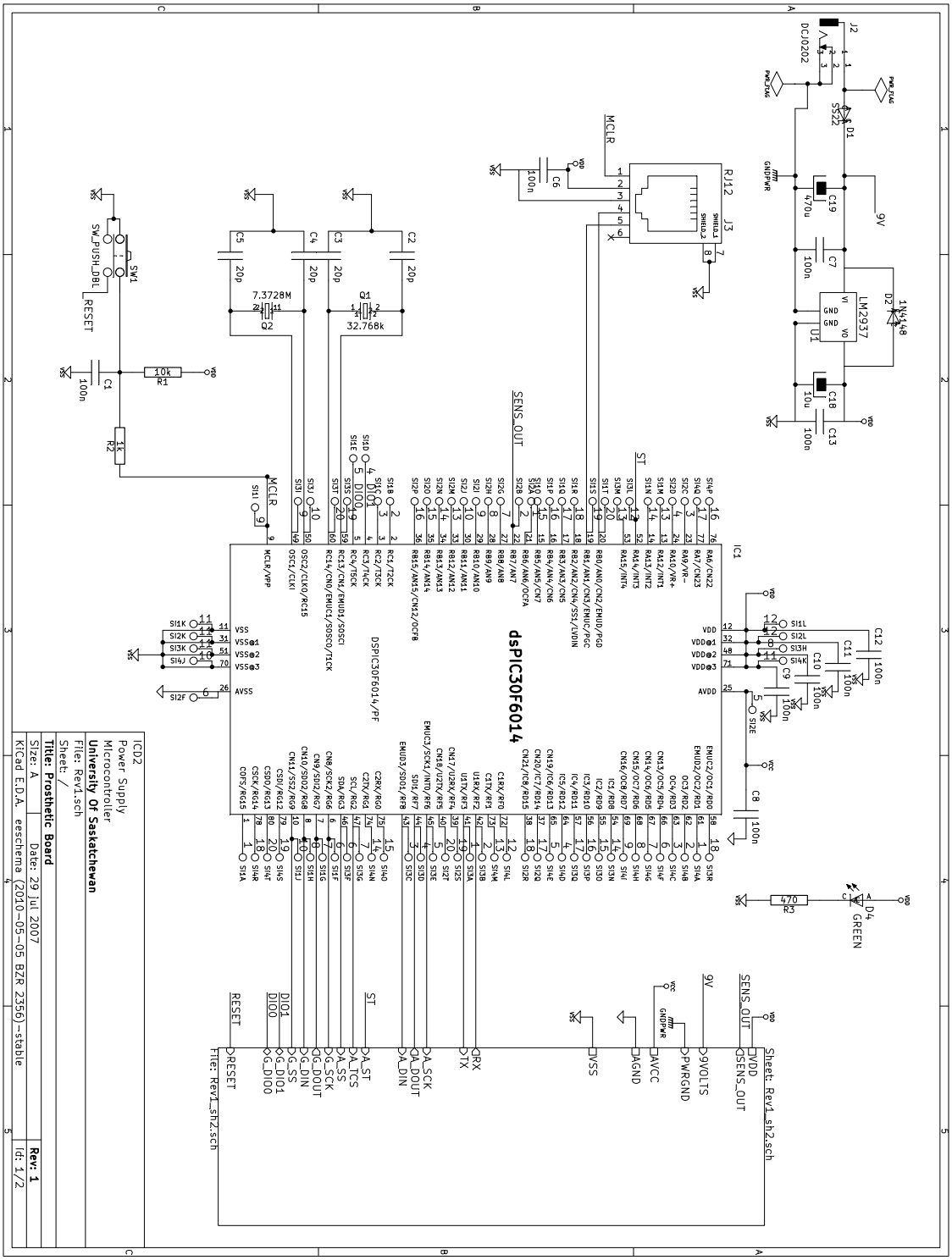

Figure A.1: Schematic for First Prototype Sensor Board

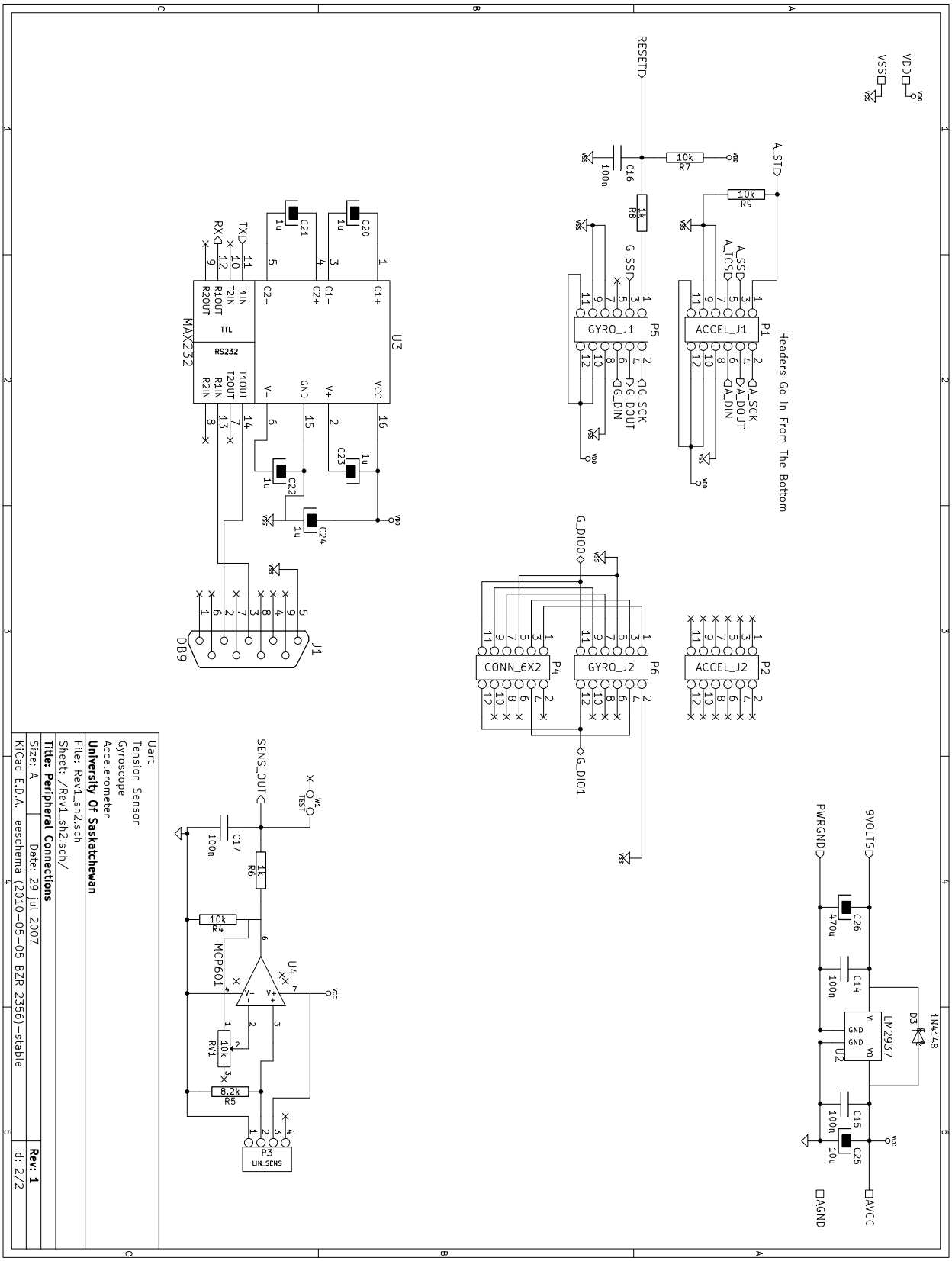

Figure A.2: Schematic for First Prototype

Appendix B

First Prototype Circuit Layout

The following are the images for the layout of the first prototype PCB as built. All images are as viewed from the top.

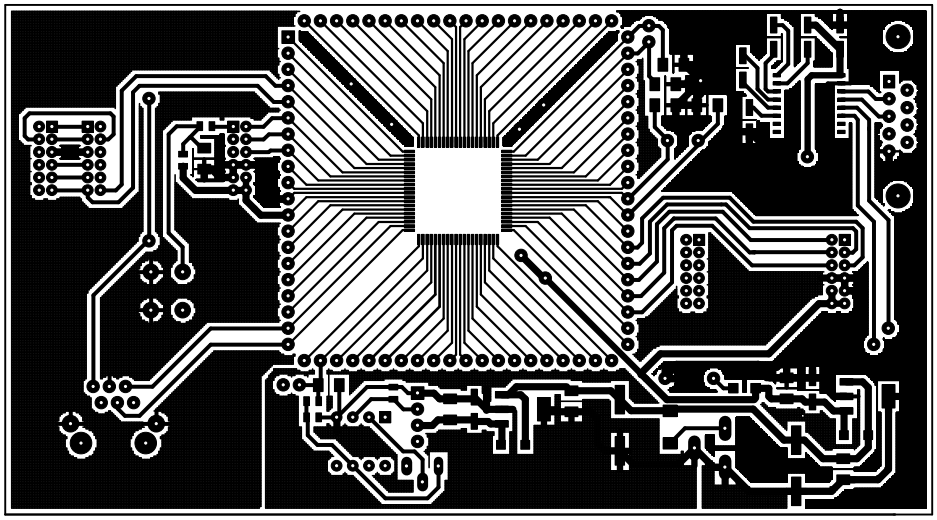

Figure B.1: PCB layout for top copper layer (First Prototype)

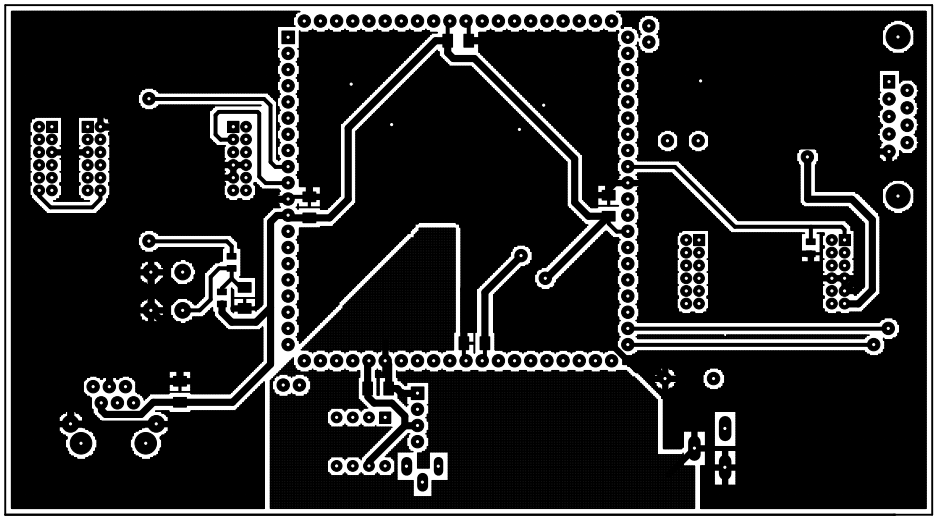

Figure B.2: PCB layout for bottom copper layer (First Prototype)

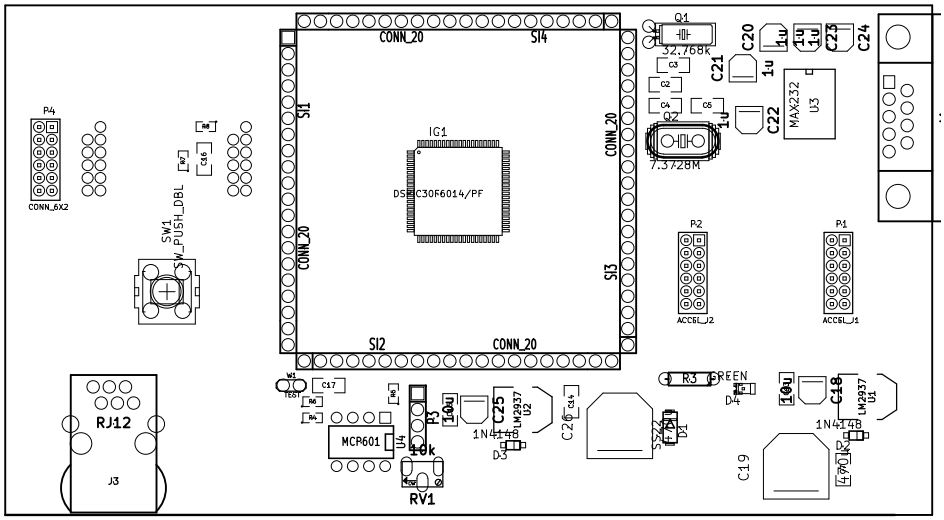

Figure B.3: PCB placement for components on the top side (First Prototype)

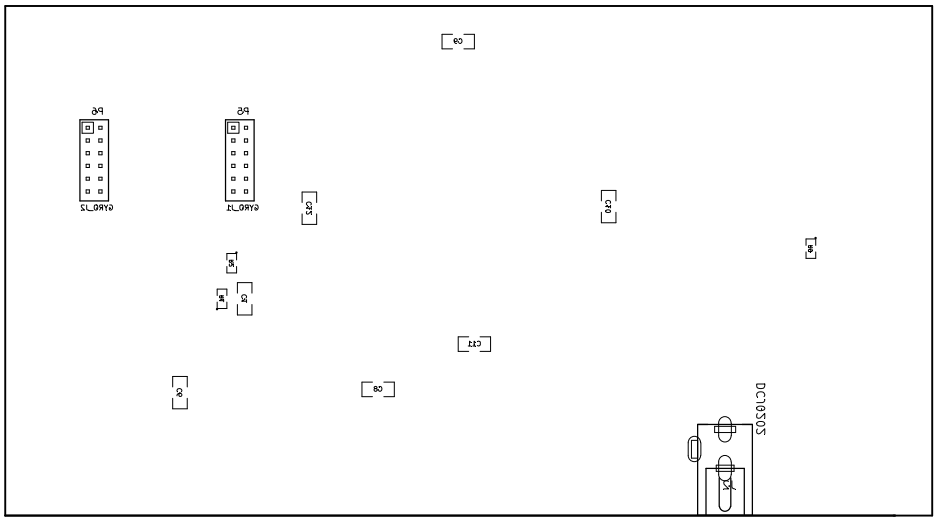

Figure B.4: PCB placement for components on the bottom side (First Prototype)

Appendix C

Second Prototype Schematic

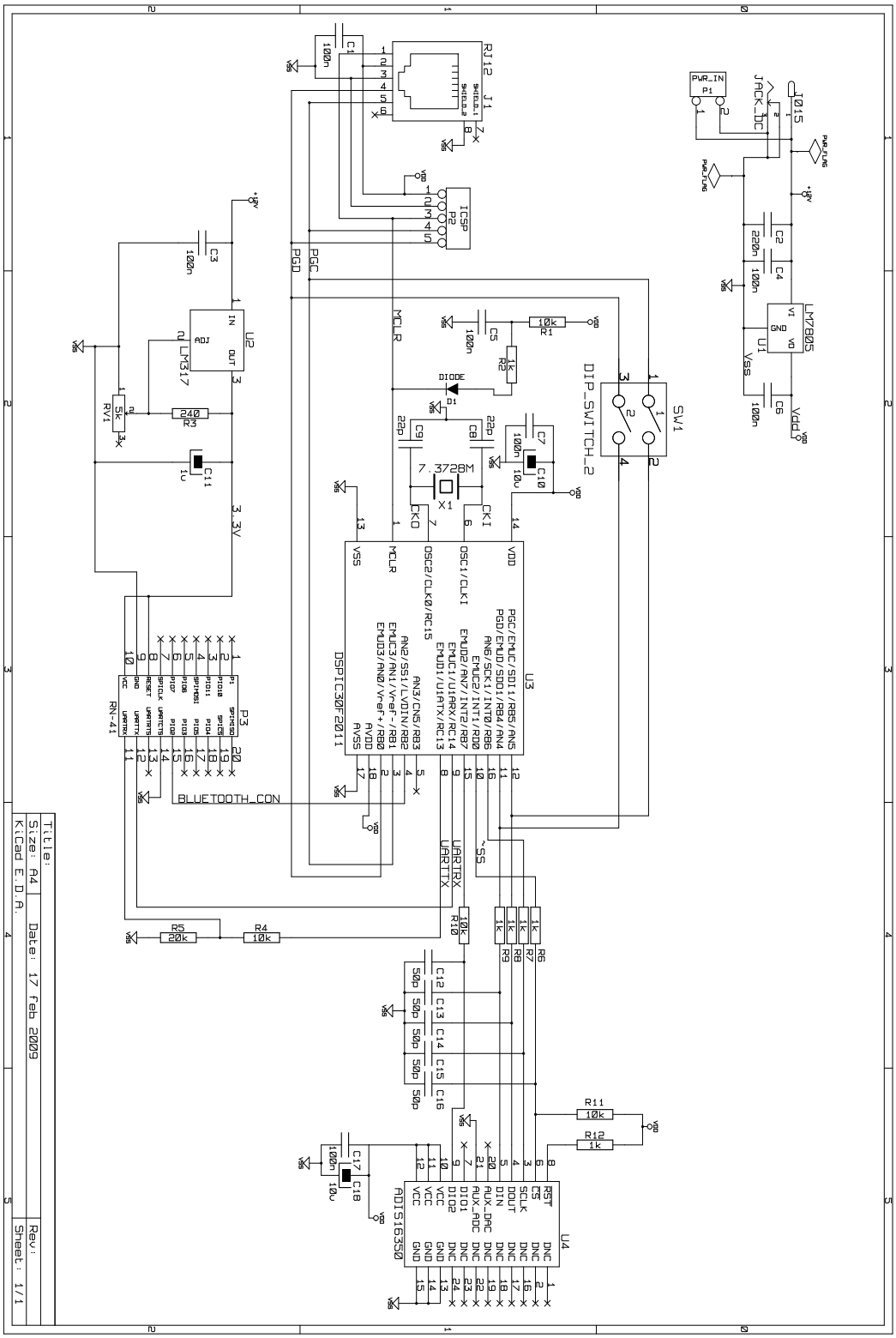

Figure C.1: Schematic for Second Prototype Sensor Board

Appendix D

Second Prototype Circuit Layout

The following are the images for the layout of the prototype PCB as built.

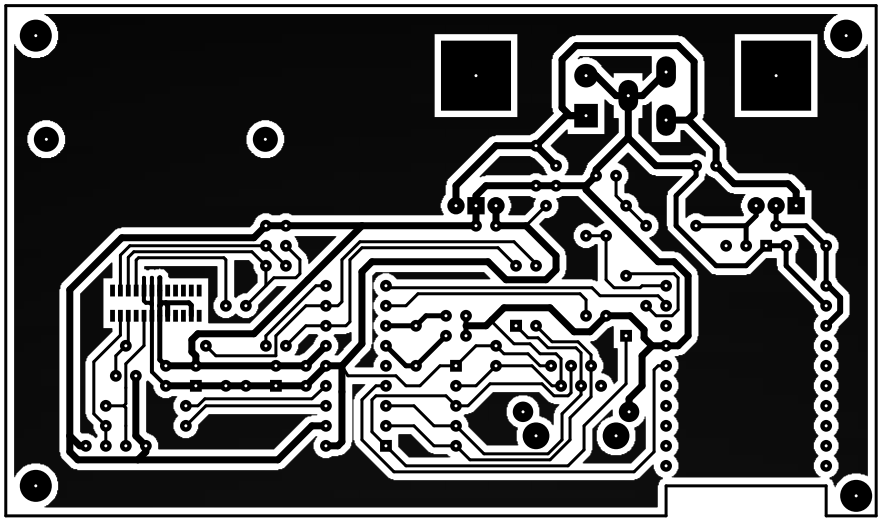

Figure D.1: PCB layout for copper layer (Second Prototype)

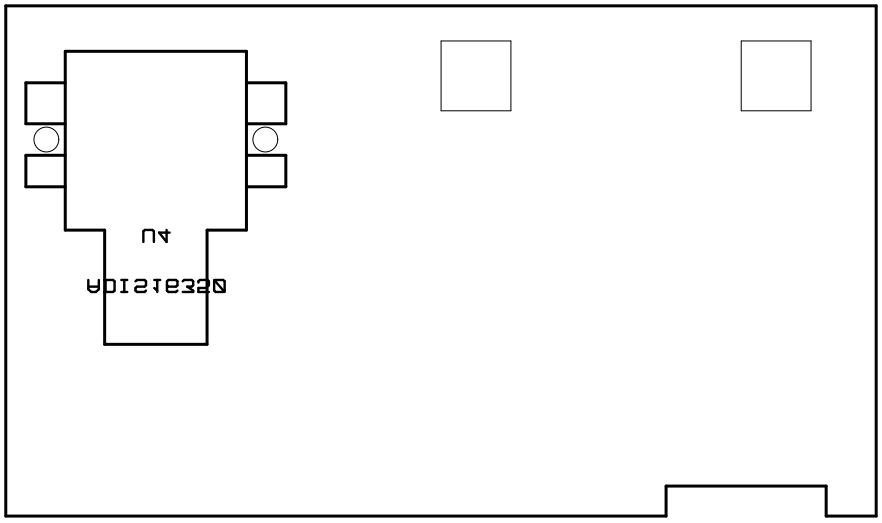

Figure D.2: PCB placement for components on the copper side (Second Prototype)

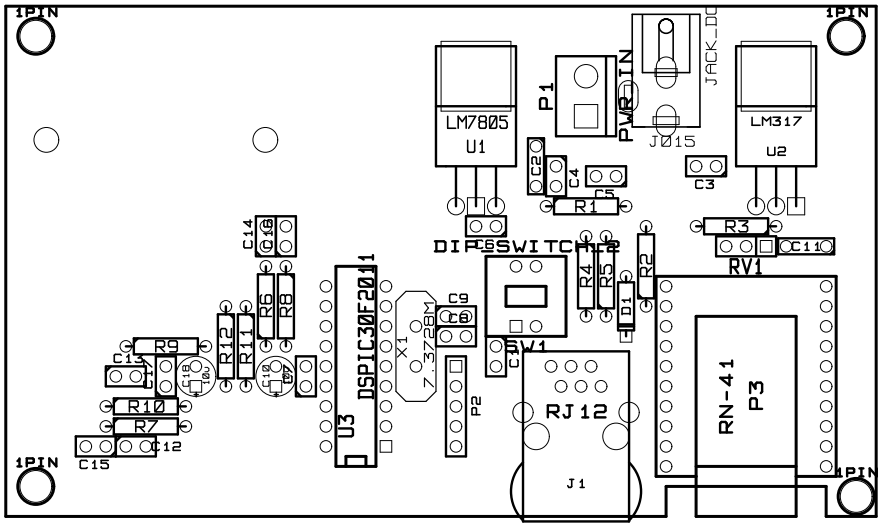

Figure D.3: PCB placement for components on the component side (Second Prototype)